NVIDIA

TECHNICAL REFERENCE MANUAL

NVIDIA Tegra X1 Mobile Processor

Abstract

The Technical Reference Manual focuses on the logical organization and control of Tegra X1 mobile processors. It provides information for those modules that interface to external devices, or those that control fundamental chip operations. The modules detailed in this document provide an overview, any necessary programming guidelines, and a register listing for that module. Internal functional units such as video and graphics hardware acceleration are controlled by NVIDIA provided software and not documented here. Refer to the software release notes for features supported in your product.

Revision History

Version	Date	Description
1.0p	NOV 09, 2015	Initial version.
1.1p	MAY 05, 2016	• Activity Monitor: BIF bit reserved. • Address Map: MIPI_BIF and NOR Flash marked "Reserved"; updated Client to Software name mapping table; added new section SMMU MC Client Address Ranges. • APB_DMA: BIF bit reserved. • Audio Processing Engine: Added paragraph to Section APE Interrupt Controller (AGIC); Added missing I2S_AXBAR registers. • Clock and Reset Controller: Added SW Default column to applicable register tables; added the following note to SDMMC4_CLK_DIVISOR, SDMMC2_CLK_DIVISOR register fields "Note: The divisor is ignored if the *_LJ options are chosen in corresponding _SRC fields"; added Clocking Sources diagram; updated Software Default values for CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0; updated description for PLLE_PTS; updated Section Clock Generation Logic for SDMMC; CLK_RST_CONTROLLER_CLK_SOURCE_MIPIBIF_0 bits reserved. • Display Controller: Updated Features section, added note to Output Capabilities, updated DC_<D/T>_WIN_<DD/TD>_COLOR_DEPTH_0 register description. • Display Interfaces: • Added note to SOR_NV_PDISP_HEAD_STATE0_0 register • and updated SOR_NV_PDISP_SOR_STATE1_0 register descriptions. • I2C Controller: Several changes throughout the chapter. • Introduction: MIPIBIF removed from diavgram. • Memory Controller: Added Sections Latency Allowances, Timestamps, Deadlines, and Slacks and Arbitration Domain. • MIPI D-PHY Calibration for CSI and DSI: Updated register description for MIPI_CAL_MIPI_BIAS_PAD_CFG2_0. • Multi-Purpose I/O Pins and Pin Multiplexing (Pinmuxing): Removed UART boot section--not applicable; updated note in PINMUX_AUX_TOUCH_CLK_0 register; Updated DRV_TYPE[1:0] description in Table: Pinmux Register Format. • Power Management Controller: Updated Secure Registers section; updated register descriptions for APBDEV_PMC_SEC_DISABLE_0, APBDEV_PMC_PMC_SWRST_0, APBDEV_PMC_BLINK_TIMER_0, APBDEV_PMC_PWRGATE_TOGGLE_0, updated APBDEV_PMC_PWR_DET_VAL_0 register bit values; reserved BIF bit. • PWM Controller: Updated Functionality section pulse width bits. • SD/MMC Controller: Updated Section 32.7.5.5 DLL and Inbound Clock Trimmer Power Supply (VREG) programming; updated values for APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_CLK_CFG_CAL_DRVUP_SLWF/_SLWR and APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC1_PAD_CAL_DRVUP/_DRVDN. • USB Controller: Updated Section: Initial Role, under Section On-The-Go (OTG) support; added note regarding wake host capability; added PLLE_PTS register bit programming information.

Version	Date	Description
1.2p	DEC 06, 2016	• APB Miscellaneous - Added register APB_MISC_GP_ASDBGREG_0. - Note added regarding programming CFG2TMC_RAM_SVOP_PDP. • Audio Processing Engine: - Marked IQC register fields as unused/reserved. - Removed duplicated AXBAR registers. • Boot and Power Management Processor-Lite: - Removed text associated with the MC/EMC frequency switch and DRAM temperature polling/ refresh adjustment. • Clock and Reset Controller - Updated Clocking Sources diagram. • CPU Complex - Added MSELECT_CONFIG register. • Design for Debugging: - Added “Managing CPU Power Down During Debug". • Display Interfaces: - Removed unsupported features (MST, DPU, and RNDGEN). - Removed TMDS Macro Configuration. - Added LC128 registers to “Enable Encryption”. - Removed DPAUX Scratch Registers. • Host Subsystem: - Added DPAUX1, TSEC1, and VII2C clients. • Interrupt Controller - Lines 75 and 76 updated in Table: Tertiary Interrupt Controller (TRI_ICTLR) Mapping. • Memory Controller - Added SWID_0/1 encoding to “Software Client-Groupings. • MIPI D-PHY Calibration for CSI and DSI: - Added Mapping of MIPI Calibration Control Registers with MIPI Pad diagram and text. - Updated register MIPI_CAL_MIPI_BIAS_PAD_CFG2_0. • Multi-Purpose I/O Pins and Pin-Multiplexing: - Updated GPIO Register Address Map table and the table in Section VGPIO Mapping. - Added note above Pad Control Groups Register Addresses. - Updated GPIO_DB_CTRL and GPIO_DB_CNT registers. - Added note to PINMUX_AUX_TOUCH_CLK_0. - Updated Pinmux setting E_OD. - Added Section on GPIOs in Loopback Mode. - Updated note in PINMUX_AUX_TOUCH_CLK_0 register. - Updated DRV_TYPE[1:0] Description in Table: Pinmux Register Format. • PCI Express Controller: - Added/revised root port registers to "NVIDIA Private Registers” at offsets 0xC10, 0xC14, 0xC18, 0xF0C, 0xF30, 0xF50, and 0xFF8. - Revised step 23a in “Root Port Initialization Sequence” • Power Management Controller: - Added content for APBDEV_PMC_NO_IOPOWER_0 register. • PWM Controller: - Added note to register PWM_CONTROLLER_PWM_CSR_0. • SD/MMC Controller: - Added “SDR50 Mode Workaround for Avoiding Read CRC Errors”. • Serial Peripheral Interface Controller: - Updated max SPI clock speed. - Added note to Section PIO Mode. - Updated Packed and Unpacked Modes Section.

Version	Date	Description
1.3p	April 23, 2019	• Clock Reset Controller: Updated registers. • CSI: Updated CSI-2 version number to v1.1. • Memory Controller: Updated bit field 15 of EMC_FBIO_CFG8_0 register. • Pixel Memory Format: Updated the Video Encode Path (NVENC) table. • SD/MMC Controller: Added Post-Tuning Procedure and updated DLL and Inbound Clock Trimmer Power Supply (VREG) Programming. • USB Complex: Updated to ensure clarity regarding isochronous endpoints are not supported.

CHAPTER 1: INTRODUCTION

The NVIDIA Tegra® X1 mobile processor is a complete applications and digital media system built around several powerful hardware elements:

- Graphics: NVIDIA® GeForce® Maxwell Graphics Processing Unit (GPU). The GPU fully supports DX11, Shader Model 4, and OpenGL4.5 as well as OpenGL ES 3.1. It supports Unified shaders and is GPU compute capable with 256 CUDA cores. The GPU supports all the same features as discrete NVIDIA GPUs, including PhysX, CUDA, OpenCL, and DX compute. It is highly power optimized for best performance in mobile and embedded use cases.
- CPU Complex: Quad ARM® Cortex®-A57 cores. The CPU cores support ARMv8, executing both 64-bit Aarch64 code, and 32-bit Aarch32 code including legacy ARMv7 applications. The Cortex-A57 processors each have 48 KB Instruction and 32 KB Data Level 1 caches; and have a 2 MB shared Level 2 unified cache.
- Memory Controller: 64-bit DRAM interface providing high bandwidth. LP-DDR4 DRAM type is supported.
- Video Decoder: The advanced video decoder supports full motion 60fps playback of 4K ultra-high-definition video with 10-bit pixels. It supports H.265, H.264, VP9, CP8 VC-1, MPEG-2, and MPEG-4 video standards and multiple audio standards with dedicated hardware.
- Video Encoder: A high performance H.265/H.264 capable hardware video encoder. This processor supports H.265 and H.264 BP/MP/HP/MVC and VP8 encoding.
- Audio Subsystem: A dedicated audio subsystem based on a Cortex-A9 processor and dedicated RAM. Full hardware support for multi-channel audio over multiple interfaces, and a broad and flexible range of audio I/O interfaces.
- Imaging: A high-quality hardware accelerated still-image and video capture path, with dual next-generation ISPs.
- Display: Dual display controllers with MIPI-DSI output, along with eDP support for LCD panels and HDMI 2.0 output for external display devices. Multiple line pixel storage allows more memory-efficient scaling operations and pixel fetching. Hardware display surface rotation is also provided for bandwidth reduction.

In addition to these major elements, Tegra X1 mobile processors have a broad range of peripheral interfaces to enable communication with wireless baseband, other communications peripherals, audio codecs, power management, and mass storage. When combined with baseband and PMIC chips, the Tegra X1 mobile processor provides the functionality needed to build a range of both low-power mobile or high-performance embedded devices. Dedicated high-performance mass storage controllers, with their own DMA engines, free the CPU Complex from routine data management tasks.

Selected SKUs also contain a companion set of quad ARM Cortex-A53 cores in a switched configuration with the A57 core complex. The Cortex-A53 processors each have 32 KB Instruction and 32 KB Data Level 1 caches; and have a 512 KB shared Level 2 unified cache.Tegra X1 Technical Reference Manual

1.1 The Role of the Technical Reference Manual

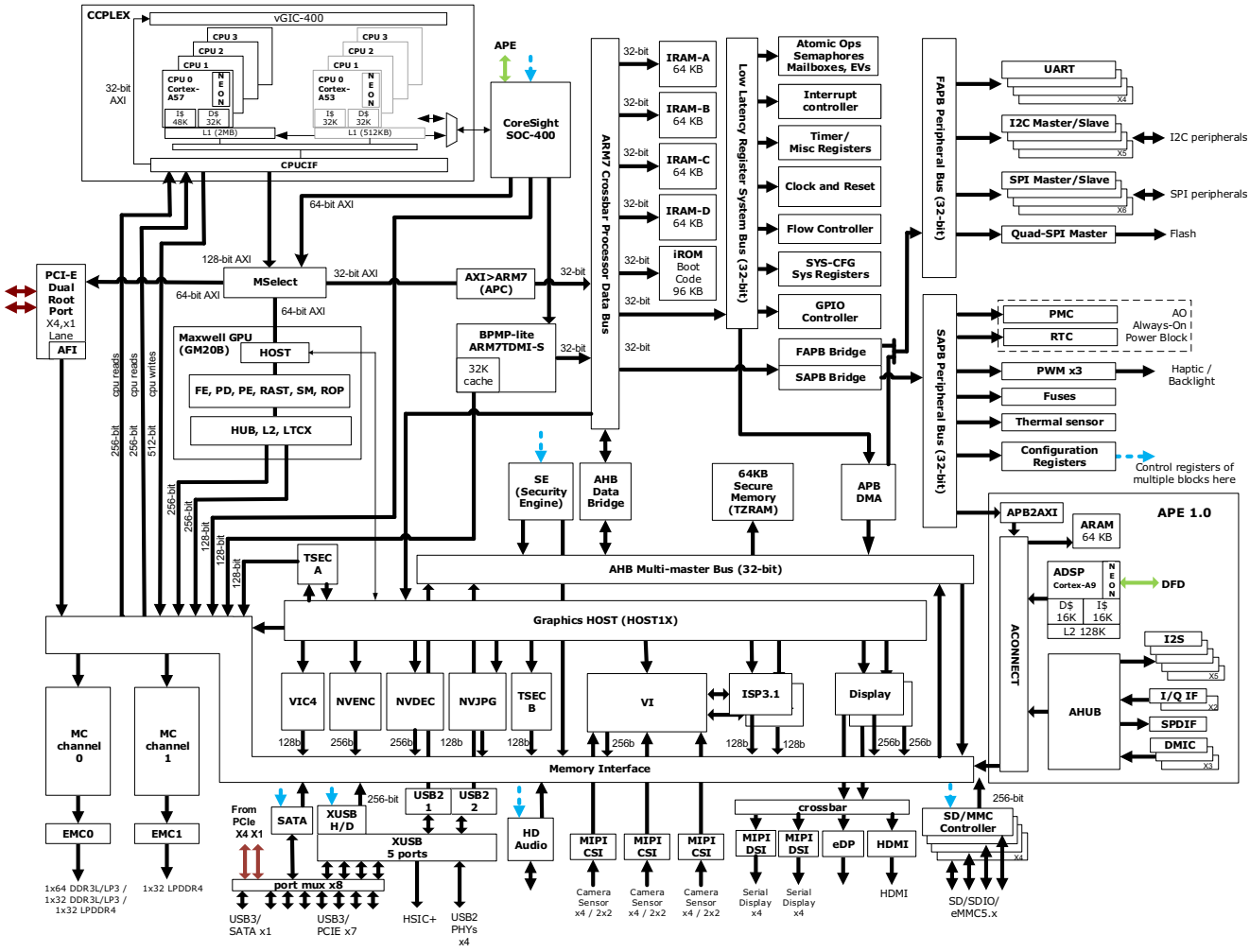
This document is intended to provide guidance to programmers writing code for Tegra X1 devices. It describes the register interfaces and hardware functionality, with the goal to aid anyone in understanding and potentially modifying the NVIDIA provided code.

It may also describe hardware functions not currently supported by NVIDIA drivers; thus the description of a capability in this document does not necessarily imply software support for that function.

1.2 Block Diagram

This diagram provides an overview of a Tegra X1 mobile processor.

Figure 1: Tegra X1 Processor Block Diagram



1.3 Memory Controller and Internal Bus Architecture

The Tegra X1 mobile processor has a highly optimized 64-bit memory controller, supporting low latency access for the CPU, optimized high bandwidth access for the graphics and video devices, and controlled latency for real time devices such as display.

There is a three-level hierarchy of memory clients:

1. Memory controller clients: The memory controller directly arbitrates between various requesters using a complex algorithm optimizing DRAM efficiency. The highest bandwidth clients all fall into this class, and they communicate directly with the memory controller using a proprietary high-speed bus.
2. AHB devices: These generally have a built-in DMA engine, and share a single memory client using the AHB bus protocol.
3. APB devices: All APB devices are slaves, and are serviced by a shared multi-channel APB DMA controller which is also an APB device.

Special provisions are made for the CPU to bypass parts of the memory controller arbitration to help achieve a lower latency.

1.4 Reading Register Tables

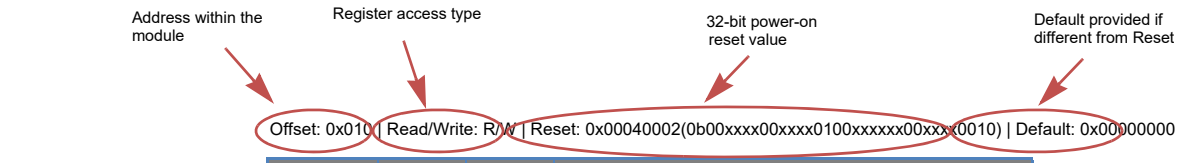
Every register table has an address line followed by a table containing the bit descriptions for that register. The address line contains:

- **Offset:** the address of the register within the specific module. Refer to the system memory map for the start address of the module, and apply the offset at the top of the table to get the register address.
- **Read/Write:** the register access type. When a register table contains the R/W column, individual bits within the register will have different R/W properties. When there is no R/W column, all bits in that register have the same R/W property. Values are RO (read only), R/W (read/write), WO (write only, and RWC (read/write clear).
- **Reset:** gives the power-on reset value in 32-bit binary. A value of x implies that the register bit has an undefined value at reset. A hexadecimal value is listed for convenience, where appropriate.
- **Default:** only displayed if the default setting is different from the Reset value.

Unspecified bits may not appear in tables (see example below). Unspecified bits should be written with their Reset values, while reads return an unknown value.

Note: *At times, a SW Default column may be present. SW Default settings are the default values that software is expected to program.*

Figure 2: Register Table Example



Unspecified bits in this example register: 31:26, 23:20, 15:10, 7:4

In the Reset field above the table, most unspecified bits are shown as 'x' in binary format. In hex format, unspecified bits within nibbles are shown as 0 in the Reset field above the table.

At power-on reset, write only the provided reset value to unspecified bits for proper operation. For example, in the Reset field above the table, bits [31:30], which are not specified in the table, must be written as 0s.

Bit	R/W	Reset	Description
25:24	RW	N1	EMEM_NUMDEV 0 = N1 1 = N2
19:16	RO	D64MB	EMEM_DEVSIZE 0 = D4 MB 1 = D8 MB 2 = D16 MB 3 = D32 MB 4 = D64 MB 5 = D128 MB 6 = D256 MB 7 = D512 MB
9:8	RW	W2	EMEM_BANKWIDTH 1 = W1 2 = W2 3 = W3
3:0	RO	W9	EMEM_COLWIDTH 0 = W7 1 = W8 2 = W9 3 = W10 4 = W11

1.5 Glossary

This glossary is intended to cover the Tegra specific acronyms used in this document; along with some others related to the ARM SOC world. Many acronyms in this document are in broad engineering use and are not documented here; we assume you already know what USB and CPU are, for example.

Term	Definition
ACPI	Advanced Configuration and Power Interface

Term	Definition
ADX	Audio Demultiplexer Block, part of the Audio Hub used to demultiplex multiple audio streams.
AHB	AMBA High-Speed Bus, a multi-master high-speed (relative to APB) bus supporting arbitration and split transactions, defined as part of AMBA 2.
AHBSLVMEM	The AHB slave used for the main memory interface. Acts as an AHB slave and provides a path from there to main memory. Refer to the AHB chapter.
AMBA	Advanced Microcontroller Bus Architecture, a set of standard buses defined by ARM.
AMX	Audio Multiplexer Block, part of the Audio Hub used to multiplex multiple audio streams together.
AO	Always ON
APB	AMBA Peripheral Bus, a simple 32-bit single master bus for peripheral devices.
APBDMA	A multi-channel DMA controller for devices on the APB bus, performing DMA between APB and AHB.
APE	Audio Processing Engine
ARM	Advanced RISC Machines, a company that licenses CPU IP to Tegra. Also: Architecture Reference Manual, so the ARM defines the CPU architecture.
AXI	AMBA Advanced eXtensible Interface, a more advanced bus than AHB defined as part of AMBA 3.
BPMP	Boot and Power Management Processor. ARM7 coprocessor used for power management related tasks at runtime.
BSEA	Bit Stream Engine for Audio applications
CAR	Clock and Reset module allows controlling clocks and resets to all the modules and subsystems in the Tegra processor.
CDE	Color Decompression Engine
CEC	Consumer Electronics Control, a part of the HDMI interface specification used for sending device control commands, often from a remote control.
COP	CO-Processor, an obsolete name for BPMP-Lite still present in legacy documentation and registers.
CSI	MIPI Camera Serial Interface, a standard high-speed serial interface for connecting cameras to the Tegra processor.
CUDA	Compute Unified Device Architecture
DMIC	Digital microphone interface, supporting direct attach of PDM microphones.
DSI	MIPI Display Serial Interface, a standard high-speed serial interface for connecting displays to the Tegra processor.
DVC	Dynamic Voltage Controller module
eDP	Embedded Display Port
EMC	External Memory Controller, a module that interfaces with external DDR/LPDDR devices.
GART	Graphic Address Relocation Table, a now obsolete mechanism for mapping from virtual to physical addresses for devices. Remains in Tegra only as an aperture to memory that may be mapped though the SMMU, the replacement for the GART.
GPIO	General Purpose Input/Output, an I/O signal uncommitted to a specific role and controlled by software.
HDMI	High-Definition Multimedia Interface, a digital connection carrying uncompressed video and audio at high speed over a single connector.
IDE	Integrated Drive Electronics (or Integrated Device Electronics)
ISP	Image Signal Processor, a hardware engine that is part of the camera processing pipeline.
MC	Memory Controller module handles requests from internal clients and arbitrates among them to allocate memory bandwidth.
MCCIF or MC-CIF	Memory Controller Client InterFace, the standard interface block between the memory controller sub-system fabric and the client device. Note that some modules may have multiple client interfaces.
MIPI	The Mobile Industry Processor Interface and industry alliance promoting a number of standard interfaces for mobile devices.

Term	Definition
MPCORE	Multi-processor CPU core, a generic term for a CPU capable of operating as part of an SMP group.
MPE	An older name for the Video Encoder in the Tegra processor capable of encoding raw video stream into MPEG. Now referred to as MSENC.
NVENC	NVIDIA Video Encoder engine.
OGL	Open Graphics Library, also known as OpenGL. An API supported on Tegra devices and accelerated in hardware by dedicated 3D and 2D engines.
PCIe	Peripheral Component Interconnect Express, a high-speed interface for external devices connected to the Tegra SOC.
PMC	Power Management Controller module controls the various power management features in the system.
POR	Power On Reset
PPSB	PortalPlayer System Bus, a proprietary register bus used for some blocks. Similar to APB. PortalPlayer is a company that was acquired by NVIDIA, and from where parts of Tegra are derived including this bus.
PWFM	Pulse Width Frequency Modulation module generates programmed pulse widths typically used to control backlight in display panels.
PVT	Process, Voltage, & Temperature
RISC	Reduced Instruction Set Computer, the CPU architecture used by ARM CPUs.
SATA	Serial Advanced Technology Attachment (ATA)
SDMMC	SD and MMC controller. An I/O controller supporting
SLINK	Serial Link, a legacy and now obsolete name for the SPI controller
SMMU	System Memory Management Unit, a block within the memory controller used to map from a virtual address space to physical addresses for device DMA.
SMP	Symmetric Multi-Processing
SOC	System On a Chip, an integrated circuit containing a CPU, memory controller and the peripheral devices needed for a computing system.
SOR	Serial Output Resource. SOR is GPU IP for driving HDMI/DP/LVDS. It converts the output of the display to a more modern high-speed serial protocol. DSI is not included since it’s not GPU IP based
S/PDIF	Sony/Phillips Digital Interconnect Format
SPI	Serial Peripheral Interface Bus, a synchronous serial data link, that operates in full duplex mode.
TSEC	Tegra Security co-processor, an embedded security processor used mainly to manage the HDCP encryption and keys on the HDMI link.
TZ	Trust Zone, a secure operating environment of the ARM CPU and the related secure parts of the SOC backbone and devices
TZRAM	Trust Zone secured RAM on the SOC.
UCQ	Unified Command Queue – a sub module within Video Decoder Engine
VDE	Video Decode Engine, a Tegra hardware acceleration block dedicated to decoding compressed video in various formats.
vGPIO	Virtual General-Purpose Input/Output
VI	Video Input block, the acronym used to describe the Tegra X1 block used for camera and related pixel input functions.
VIC	Video Image Compositer, a Tegra X1 block that implements video post-processing functions needed by a video playback application to produce the final image for the player window.
YCbCr	An alternative pixel representation that takes advantage of the properties of the human psycho-perceptual vision system. It consists of a luminance channel Y and two color difference signals Cb and Cr.
YUV	See YCbCr. U and V are equivalent to Cb and Cr, respectively.

CHAPTER 2: ADDRESS MAP

Note: *Some subsystem address maps have been separated out of the System Address Map into their own sections.*

2.1 System Address Map

Table 1: System Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
Remap	0100:0000	3fff:ffff			1008 MB
PCIe	0100:0000	3fff:ffff	0000:0000	3eff:ffff	1008 MB
PCIE_A1	0100:0000	01ff:ffff			16 MB
PCIE_A2	0200:0000	1fff:ffff			480 MB
PCIE_A3	2000:0000	3fff:ffff			512 MB
Data Memory	4000:0000	40ff:ffff			16 MB
iRAM-A	4000:0000	4000:ffff	0000:0000	0000:ffff	64 KB
iRAM-B	4001:0000	4001:ffff	0001:0000	0001:ffff	64 KB
iRAM-C	4002:0000	4002:ffff	0002:0000	0002:ffff	64 KB
iRAM-D	4003:0000	4003:ffff	0003:0000	0003:ffff	64 KB
NOR Flash	Reserved				
Graphics Host Registers	5000:0000	5003:ffff			256 KB
Host1x	5000:0000	5003:ffff	0000:0000	0003:ffff	256 KB
ARM registers (@ PERIPHBASE)	5004:0000	5005:ffff			128 KB
ARM PERIPHBASE	5004:0000	5005:ffff	0000:0000	0001:ffff	128 KB
ARM Interrupt Distributor	5004:1000	5004:1fff	0000:1000	0000:1fff	4 KB
BPMP-Lite CACHE	5004:0000	5004:1fff	0000:0000	0000:1fff	8 KB
Interrupt Controller Physical CPU interface	5004:2000	5004:3fff	0000:2000	0000:3fff	8 KB
Interrupt Controller Virtual CPU interface (Hypervisor view for requesting CPU)	5004:4000	5004:4fff	0000:4000	0000:4fff	4 KB
Interrupt Controller Virtual CPU interface (Hypervisor view for all CPUs)	5004:5000	5004:5fff	0000:5000	0000:5fff	4 KB
Interrupt Controller Virtual CPU interface (Virtual Machine view)	5004:6000	5004:7fff	0000:6000	0000:7fff	8 KB
MSelect	5006:0000	5006:0fff			4 KB
MSelect Register Space	5006:0000	5006:0fff	0000:0000	0000:0fff	4 KB
Graphics Host	5400:0000	54ff:ffff			16 MB
VI	5408:0000	540b:ffff	0008:0000	000b:ffff	256 KB
CSI	5408:0000	540b:ffff	0008:0000	000b:ffff	256 KB
ISP	5460:0000	5463:ffff	0060:0000	0063:ffff	256 KB
ISPB	5468:0000	546b:ffff	0068:0000	006b:ffff	256 KB
VII2C	546c:0000	546f:ffff	006c:0000	006f:ffff	256 KB
Display A	5420:0000	5423:ffff	0020:0000	0023:ffff	256 KB
Display B	5424:0000	5427:ffff	0024:0000	0027:ffff	256 KB
DSI	5430:0000	5433:ffff	0030:0000	0033:ffff	256 KB
VIC	5434:0000	5437:ffff	0034:0000	0037:ffff	256 KB
NVENC	544c:0000	544f:ffff	004c:0000	004f:ffff	256 KB

Table 1: System Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
NVDEC	5448:0000	544b:ffff	0048:0000	004b:ffff	256 KB
NVJPG	5438:0000	543b:ffff	0038:0000	003b:ffff	256 KB
DSIB	5440:0000	5443:ffff	0040:0000	0043:ffff	256 KB
TSEC	5450:0000	5453:ffff	0050:0000	0053:ffff	256 KB
TSEC2	5410:0000	5413:ffff	0010:0000	0013:ffff	256 KB
SOR	5454:0000	5457:ffff	0054:0000	0057:ffff	256 KB
SOR1	5458:0000	545b:ffff	0058:0000	005b:ffff	256 KB
DPAUX	545c:0000	545f:ffff	005c:0000	005f:ffff	256 KB
DPAUX1	5404:0000	5407:ffff	0004:0000	0007:ffff	256 KB
GPU	5700:0000	5fff:ffff	0000:0000	08ff:ffff	144 MB
GPU_GART	5700:0000	5fff:ffff			144 MB
PPSB	6000:0000	60ff:ffff			16 MB
uP-TAG	6000:0000	6000:0fff			4 KB
Resource Semaphore	6000:1000	6000:1fff	0000:1000	0000:1fff	4 KB
Arbitration Semaphore	6000:2000	6000:2fff	0000:2000	0000:2fff	4 KB
ARB-PRI	6000:3000	6000:3fff	0000:3000	0000:3fff	4 KB
Primary ICTLR	6000:4000	6000:403f	0000:4000	0000:403f	64 B
Primary ICTLR ARB-GNT	6000:4040	6000:40ff	0000:4040	0000:40ff	192 B
Secondary ICTLR	6000:4100	6000:41ff	0000:4100	0000:41ff	256 B
Tertiary ICTLR	6000:4200	6000:42ff	0000:4200	0000:42ff	256 B
Quaternary ICTLR	6000:4300	6000:43ff	0000:4300	0000:43ff	256 B
Quinary ICTLR	6000:4400	6000:44ff	0000:4400	0000:44ff	256 B
Senary ICTLR	6000:4500	6000:45ff	0000:4500	0000:45ff	256 B
Common ICTLR	6000:4800	6000:48ff	0000:4800	0000:48ff	256 B
TMR	6000:5000	6000:53ff	0000:5000	0000:53ff	1 KB
TMR1			0000:0000	0000:0007	8 B
TMR2			0000:0008	0000:000f	8 B
TMRUS			0000:0010	0000:004f	64 B
TMR3			0000:0050	0000:0057	8 B
TMR4			0000:0058	0000:005f	8 B
TMR5			0000:0060	0000:0067	8 B
TMR6			0000:0068	0000:006f	8 B
TMR7			0000:0070	0000:0077	8 B
TMR8			0000:0078	0000:007f	8 B
TMR9			0000:0080	0000:0087	8 B
TMR0			0000:0088	0000:008f	8 B
TMR10			0000:0090	0000:0097	8 B
TMR11			0000:0098	0000:009f	8 B
TMR12			0000:00a0	0000:00a7	8 B
TMR13			0000:00a8	0000:00af	8 B
WDT0			0000:0100	0000:011f	32 B
WDT1			0000:0120	0000:013f	32 B
WDT2			0000:0140	0000:015f	32 B
WDT3			0000:0160	0000:017f	32 B
WDT4			0000:0180	0000:019f	32 B
TMR_SHARED			0000:01a0	0000:01bf	32 B

Table 1: System Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
Clock and Reset (CAR)	6000:6000	6000:6fff	0000:6000	0000:6fff	4 KB
Flow Controller	6000:7000	6000:7fff	0000:7000	0000:7fff	4 KB
AHB-DMA	6000:8000	6000:9fff	0000:8000	0000:9fff	8 KB
AHB-DMA CH0	6000:9000	6000:901f	0000:9000	0000:901f	32 B
AHB-DMA CH1	6000:9020	6000:903f	0000:9020	0000:903f	32 B
AHB-DMA CH2	6000:9040	6000:905f	0000:9040	0000:905f	32 B
AHB-DMA CH3	6000:9060	6000:907f	0000:9060	0000:907f	32 B
APB-DMA	6002:0000	6002:3fff	0002:0000	0002:3fff	16 KB
APB-DMA CH0	6002:1000	6002:103f	0002:1000	0002:103f	64 B
APB-DMA CH1	6002:1040	6002:107f	0002:1040	0002:107f	64 B
APB-DMA CH2	6002:1080	6002:10bf	0002:1080	0002:10bf	64 B
APB-DMA CH3	6002:10c0	6002:10ff	0002:10c0	0002:10ff	64 B
APB-DMA CH4	6002:1100	6002:113f	0002:1100	0002:113f	64 B
APB-DMA CH5	6002:1140	6002:117f	0002:1140	0002:117f	64 B
APB-DMA CH6	6002:1180	6002:11bf	0002:1180	0002:11bf	64 B
APB-DMA CH7	6002:11c0	6002:11ff	0002:11c0	0002:11ff	64 B
APB-DMA CH8	6002:1200	6002:123f	0002:1200	0002:123f	64 B
APB-DMA CH9	6002:1240	6002:127f	0002:1240	0002:127f	64 B
APB-DMA CH10	6002:1280	6002:12bf	0002:1280	0002:12bf	64 B
APB-DMA CH11	6002:12c0	6002:12ff	0002:12c0	0002:12ff	64 B
APB-DMA CH12	6002:1300	6002:133f	0002:1300	0002:133f	64 B
APB-DMA CH13	6002:1340	6002:137f	0002:1340	0002:137f	64 B
APB-DMA CH14	6002:1380	6002:13bf	0002:1380	0002:13bf	64 B
APB-DMA CH15	6002:13c0	6002:13ff	0002:13c0	0002:13ff	64 B
APB-DMA CH16	6002:1400	6002:143f	0002:1400	0002:143f	64 B
APB-DMA CH17	6002:1440	6002:147f	0002:1440	0002:147f	64 B
APB-DMA CH18	6002:1480	6002:14bf	0002:1480	0002:14bf	64 B
APB-DMA CH19	6002:14c0	6002:14ff	0002:14c0	0002:14ff	64 B
APB-DMA CH20	6002:1500	6002:153f	0002:1500	0002:153f	64 B
APB-DMA CH21	6002:1540	6002:157f	0002:1540	0002:157f	64 B
APB-DMA CH22	6002:1580	6002:15bf	0002:1580	0002:15bf	64 B
APB-DMA CH23	6002:15c0	6002:15ff	0002:15c0	0002:15ff	64 B
APB-DMA CH24	6002:1600	6002:163f	0002:1600	0002:163f	64 B
APB-DMA CH25	6002:1640	6002:167f	0002:1640	0002:167f	64 B
APB-DMA CH26	6002:1680	6002:16bf	0002:1680	0002:16bf	64 B
APB-DMA CH27	6002:16c0	6002:16ff	0002:16c0	0002:16ff	64 B
APB-DMA CH28	6002:1700	6002:173f	0002:1700	0002:173f	64 B
APB-DMA CH29	6002:1740	6002:177f	0002:1740	0002:177f	64 B
APB-DMA CH30	6002:1780	6002:17bf	0002:1780	0002:17bf	64 B
APB-DMA CH31	6002:17c0	6002:17ff	0002:17c0	0002:17ff	64 B
System Registers	6000:c000	6000:c2ff	0000:c000	0000:c2ff	768 B
AHB Arbitration + Gizmo Controller			0000:0000	0000:014f	336 B
AHB/APB Debug Bus			0000:0150	0000:01ff	176 B
Secure Boot			0000:0200	0000:02ff	256 B
Activity Monitor	6000:c800	6000:cbff	0000:c800	0000:cbff	1 KB
GPIO-1	6000:d000	6000:d0ff	0000:d000	0000:d0ff	256 B

Table 1: System Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
GPIO-2	6000:d100	6000:d1ff	0000:d100	0000:d1ff	256 B
GPIO-3	6000:d200	6000:d2ff	0000:d200	0000:d2ff	256 B
GPIO-4	6000:d300	6000:d3ff	0000:d300	0000:d3ff	256 B
GPIO-5	6000:d400	6000:d4ff	0000:d400	0000:d4ff	256 B
GPIO-6	6000:d500	6000:d5ff	0000:d500	0000:d5ff	256 B
GPIO-7	6000:d600	6000:d6ff	0000:d600	0000:d6ff	256 B
GPIO-8	6000:d700	6000:d7ff	0000:d700	0000:d7ff	256 B
Exception vectors	6000:f000	6000:ffff	0000:f000	0000:ffff	4 KB
IPATCH	6001:dc00	6001:dfff	0001:dc00	0001:dfff	1 KB
VGPIO	6002:4000	6002:43ff	0002:4000	0002:43ff	1 KB
APB	7000:0000	73ff:ffff			64 MB
MISC	7000:0000	7000:3fff	0000:0000	0000:3fff	16 KB
PP			0000:0000	0000:03ff	1 KB
SC1X_PADS			0000:0400	0000:07ff	1 KB
GP			0000:0800	0000:0bff	1 KB
SECURE_REGS			0000:0c00	0000:0cff	256 B
OBS			0000:0d00	0000:0dff	256 B
PROTO			0000:0e00	0000:0fff	512 B
USB_AUX			0000:1000	0000:10ff	256 B
SATA_AUX			0000:1100	0000:11ff	256 B
PINMUX_AUX			0000:3000	0000:3fff	4 KB
UART-A	7000:6000	7000:603f	0000:6000	0000:603f	64 B
UART-B	7000:6040	7000:607f	0000:6040	0000:607f	64 B
UART-C	7000:6200	7000:62ff	0000:6200	0000:62ff	256 B
UART-D	7000:6300	7000:63ff	0000:6300	0000:63ff	256 B
UART-E	7000:6400	7000:64ff	0000:6400	0000:64ff	256 B
HDMI_IOBIST	7000:6500	7000:65ff	0000:6500	0000:65ff	256 B
MIPI_IOBIST	7000:6600	7000:66ff	0000:6600	0000:66ff	256 B
LPDDR2_IOBIST	7000:6700	7000:67ff	0000:6700	0000:67ff	256 B
PCIE_X2_0_IOBIST	7000:6800	7000:68ff	0000:6800	0000:68ff	256 B
PCIE_X2_1_IOBIST	7000:6900	7000:69ff	0000:6900	0000:69ff	256 B
PCIE_X4_IOBIST	7000:6a00	7000:6aff	0000:6a00	0000:6aff	256 B
SATA_IOBIST	7000:6c00	7000:6dff	0000:6c00	0000:6dff	512 B
PWM Controller	7000:a000	7000:a0ff	0000:a000	0000:a0ff	256 B
I2C	7000:c000	7000:c0ff	0000:c000	0000:c0ff	256 B
DTV	Reserved				
I2C2	7000:c400	7000:c4ff	0000:c400	0000:c4ff	256 B
I2C3	7000:c500	7000:c5ff	0000:c500	0000:c5ff	256 B
I2C4	7000:c700	7000:c7ff	0000:c700	0000:c7ff	256 B
I2C5	7000:d000	7000:d0ff	0000:d000	0000:d0ff	256 B
I2C6	7000:d100	7000:d1ff	0000:d100	0000:d1ff	256 B
SPI 2B-1	7000:d400	7000:d5ff	0000:d400	0000:d5ff	512 B
SPI 2B-2	7000:d600	7000:d7ff	0000:d600	0000:d7ff	512 B
SPI 2B-3	7000:d800	7000:d9ff	0000:d800	0000:d9ff	512 B
SPI 2B-4	7000:da00	7000:dbff	0000:da00	0000:dbff	512 B
RTC	7000:e000	7000:e0ff	0000:e000	0000:e0ff	256 B

Table 1: System Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
PMC	7000:e400	7000:efff	0000:e400	0000:efff	3 KB
FUSE	7000:f800	7000:fbff	0000:f800	0000:fbff	1 KB
KFUSE	7000:fc00	7000:ffff	0000:fc00	0000:ffff	1 KB
LA (NV debug only-unauthorized access can affect device security)	7001:0000	7001:1fff	0001:0000	0001:1fff	8 KB
SE	7001:2000	7001:3fff	0001:2000	0001:3fff	8 KB
TSENSOR	7001:4000	7001:4fff	0001:4000	0001:4fff	4 KB
CEC	7001:5000	7001:5fff	0001:5000	0001:5fff	4 KB
ATOMICS	7001:6000	7001:7fff	0001:6000	0001:7fff	8 KB
MC	7001:9000	7001:9fff	0001:9000	0001:9fff	4 KB
EMC	7001:b000	7001:bfff	0001:b000	0001:bfff	4 KB
EMCB	7001:b000	7001:bfff	0001:b000	0001:bfff	4 KB
MC0	7001:c000	7001:cfff	0001:c000	0001:cfff	4 KB
MC1	7001:d000	7001:dfff	0001:d000	0001:dfff	4 KB
EMC0	7001:e000	7001:efff	0001:e000	0001:efff	4 KB
EMC1	7001:f000	7001:ffff	0001:f000	0001:ffff	4 KB
SATA	7002:0000	7002:ffff	0002:0000	0002:ffff	64 KB
HDA	7003:0000	7003:ffff	0003:0000	0003:ffff	64 KB
APE	702c:0000	702f:ffff	002c:0000	002f:ffff	256 KB
XUSB_PADCTL	7009:f000	7009:ffff	0009:f000	0009:ffff	4 KB
XUSB_HOST	7009:0000	7009:9fff	0009:0000	0009:9fff	40 KB
XUSB_DEV	700d:0000	700d:9fff	000d:0000	000d:9fff	40 KB
DDS	700a:0000	700a:11ff	000a:0000	000a:11ff	4608 B
SDMMC-1	700b:0000	700b:01ff	000b:0000	000b:01ff	512 B
SDMMC-1B	700b:1000	700b:11ff	000b:1000	000b:11ff	512 B
SDMMC-2	700b:0200	700b:03ff	000b:0200	000b:03ff	512 B
SDMMC-2B	700b:2200	700b:23ff	000b:2200	000b:23ff	512 B
SDMMC-3	700b:0400	700b:05ff	000b:0400	000b:05ff	512 B
SDMMC-3B	700b:3400	700b:35ff	000b:3400	000b:35ff	512 B
SDMMC-4	700b:0600	700b:07ff	000b:0600	000b:07ff	512 B
SDMMC-4B	700b:4600	700b:47ff	000b:4600	000b:47ff	512 B
SPEEDO	700c:0000	700c:7fff	000c:0000	000c:7fff	32 KB
SPEEDO_0			0000:0000	0000:00ff	256 B
SPEEDO_1			0000:0100	0000:01ff	256 B
SPEEDO_PMON	700c:8000	700c:ffff	000c:8000	000c:ffff	32 KB
SPEEDO_PMON_0			0000:0000	0000:01ff	512 B
SPEEDO_PMON_1			0000:0200	0000:03ff	512 B
SYSCTR0	700f:0000	700f:ffff	000f:0000	000f:ffff	64 KB
SYSCTR1	7010:0000	7010:ffff	0010:0000	0010:ffff	64 KB
DP2	700e:0000	700e:00ff	000e:0000	000e:00ff	256 B
APB2JTAG	700e:1000	700e:11ff	000e:1000	000e:11ff	512 B
SOC_THERM	700e:2000	700e:2fff	000e:2000	000e:2fff	4 KB
MIPI_CAL	700e:3000	700e:30ff	000e:3000	000e:30ff	256 B
DVFS	7011:0000	7011:03ff	0011:0000	0011:03ff	1 KB
MIPIBIF	Reserved				
CLUSTER_CLOCK	7004:0000	7007:ffff	0004:0000	0007:ffff	256 KB
QSPI	7041:0000	7041:0fff	0041:0000	0041:0fff	4 KB

Table 1: System Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
STM	7100:0000	71ff:ffff	0100:0000	01ff:ffff	16 MB
CSITE	7200:0000	73ff:ffff	0200:0000	03ff:ffff	32 MB
AHB_A1	7800:0000	78ff:ffff			16 MB
AHB_A2	7c00:0000	7dff:ffff			32 MB
PPCS (AHB to MC flush)	7c00:0000	7c00:ffff	0000:0000	0000:ffff	64 KB
TZRAM	7c01:0000	7c01:ffff	0001:0000	0001:ffff	64 KB
USB	7d00:0000	7d00:17ff	0100:0000	0100:17ff	6 KB
USB2	7d00:4000	7d00:57ff	0100:4000	0100:57ff	6 KB
EMEM_LO	8000:0000	ffff:ffff			2 GB
EMEM_HI	0000:0000	0000:0002			3 B
External Memory	8000:0000	27fff:ffff			8 GB
EMEM	8000:0000	27fff:ffff	0000:0000	1ffff:ffff	8 GB
IROM (boot code)	0010:0000	0011:7fff			96 KB
IROML	0010:0000	0011:7fff			96 KB
Lo-VEC	0000:0000	0000:ffff			64 KB
IROM_LOVEC	0000:0000	00ff:ffff			16 MB

2.2 APE Address Map

Table 2: APE Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
APE Register Base	0000:0000	ffff:ffff	0000:0000	004:0000	4 GB
AROM	0000:0000	003F:FFFF	0000:0000	003F:FFFF	3 MB
ARAM	0040:0000	00BF:FFFF	0040:0000	00BF:FFFF	8 MB
ADSP_PERIPHRegister	00C0:0000	00C0:1FFF	00C0:0000	00C0:1FFF	-
ADSP_L2CCRegister	00C0:2000	00C0:2FFF	00C0:2000	00C0:2FFF	-
DRAM1	0100:0000	702B:FFFF	0100:0000	702B:FFFF	1264 MB
AHOST	702C:0000	702E:BFFF	0000:0000	0002:BFFF	176 KB
AMISCRegister	702E:C000	702E:DFFF	0002:C000	0002:DFFF	8 KB
ABRIDGERegister	702E:E000	702E:EEFF	0002:E000	0002:EEFF	3840 KB
AASRegister	702E:EF00	702E:EFFF	0002:EF00	0002:EFFF	256 KB
AMCRegister	702E:F000	702E:FFFF	0002:F000	0002:FFFF	4 KB
ACONNECTRegister	702F:0000	702F:7FFF	0003:0000	0003:7FFF	32 KB
AGICRegister	702F:8000	702F:FFFF	0003:8000	0003:FFFF	32 KB
DRAM2	7030:0000	FFFF:FFFF	7030:0000	FFFF:FFFF	2.25 GB
AHOST_CH0	702c:0000	702c:0FFF	0000:0000	0000:0FFF	4 KB
AHOST_CH1	702c:1000	702c:1FFF	0000:1000	0000:1FFF	4 KB
AHOST_CH2	702c:2000	702c:2FFF	0000:2000	0000:2FFF	4 KB
AHOST_CH3	702c:3000	702c:3FFF	0000:3000	0000:3FFF	4 KB
AHOST_CH4	702c:4000	702c:4FFF	0000:4000	0000:4FFF	4 KB
AHOST_CH5	702c:5000	702c:5FFF	0000:5000	0000:5FFF	4 KB
AHOST_CH6	702c:6000	702c:6FFF	0000:6000	0000:6FFF	4 KB
AHOST_CH7	702c:7000	702c:7FFF	0000:7000	0000:7FFF	4 KB
RESERVED	702c:8000	702c:FFFF	0000:8000	0001:DFFF	32 KB
AHUB	702d:0000	702d:FFFF	0001:0000	0001:FFFF	4 GB

Table 2: APE Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
ADMAIF	702d:0000	702d:07FF	0001:0000	0001:07ff	2 KB
AXBAR	702d:0800	702d:0FFF	0001:0800	0001:0fff	2 KB
I2S	702d:1000	702d:1FFF	0001:1000	0001:1fff	4 KB
I2S1	702d:1000	702d:10FF	0001:1000	0001:10ff	256 Bytes
I2S2	702d:1100	702d:11FF	0001:1100	0001:11ff	256 Bytes
I2S3	702d:1200	702d:12FF	0001:1200	0001:12ff	256 Bytes
I2S4	702d:1300	702d:13FF	0001:1300	0001:13ff	256 Bytes
I2S5	702d:1400	702d:14FF	0001:1400	0001:14ff	256 Bytes
SFC	702d:2000	702d:2FFF	0001:2000	0001:2fff	4 KB
SFC1	702d:2000	702d:21FF	0001:2000	0001:21ff	512 Bytes
SFC2	702d:2200	702d:23FF	0001:2200	0001:23ff	512 Bytes
SFC3	702d:2400	702d:25FF	0001:2400	0001:25ff	512 Bytes
SFC4	702d:2600	702d:27FF	0001:2600	0001:27ff	512 Bytes
AMX	702d:3000	702d:37FF	0001:3000	0001:37ff	2 KB
AMX1	702d:3000	702d:30FF	0001:3000	0001:30ff	256 Bytes
AMX2	702d:3100	702d:31FF	0001:3100	0001:31ff	256 Bytes
ADX	702d:3800	702d:3FFF	0001:3800	0001:3fff	2 KB
ADX1	702d:3800	702d:38FF	0001:3800	0001:38ff	256 Bytes
ADX2	702d:3900	702d:39FF	0001:3900	0001:39ff	256 Bytes
DMIC	702d:4000	702d:4FFF	0001:4000	0001:4fff	4 KB
DMIC1	702d:4000	702d:40FF	0001:4000	0001:40ff	256 Bytes
DMIC2	702d:4100	702d:41FF	0001:4100	0001:41ff	256 Bytes
DMIC3	702d:4200	702d:42FF	0001:4200	0001:42ff	256 Bytes
DSPK	702d:5000	702d:5FFF	0001:5000	0001:5fff	4 KB
DSPK1	702d:5000	702d:50FF	0001:5000	0001:50ff	256 Bytes
DSPK2	702d:5100	702d:51FF	0001:5100	0001:51ff	256 Bytes
SPDIF	702d:6000	702d:61FF	0001:6000	0001:61ff	512 Bytes
SPDIF1	702d:6000	702d:61FF	0001:6000	0001:61ff	512 Bytes
AFC	702d:7000	702d:7FFF	0001:7000	0001:7fff	4 KB
AFC1	702d:7000	702d:70FF	0001:7000	0001:70ff	256 Bytes
AFC2	702d:7100	702d:71FF	0001:7100	0001:71ff	256 Bytes
AFC3	702d:7200	702d:72FF	0001:7200	0001:72ff	256 Bytes
AFC4	702d:7300	702d:73FF	0001:7300	0001:73ff	256 Bytes
AFC5	702d:7400	702d:74FF	0001:7400	0001:74ff	256 Bytes
AFC6	702d:7500	702d:75FF	0001:7500	0001:75ff	256 Bytes
OPE1	702d:8000	702d:83FF	0001:8000	0001:83ff	1 KB
OPE1_COMMON	702d:8000	702d:80FF	0001:8000	0001:80FF	256 Bytes
PEQ1	702d:8100	702d:81FF	0001:8100	0001:81FF	256 Bytes
MBDRC1	702d:8200	702d:83FF	0001:8200	0001:83FF	512Bytes
OPE2	702d:8400	702d:87FF	0001:8400	0001:87FF	1 KB
OPE2_COMMON	702d:8400	702d:84FF	0001:8400	0001:84FF	256 Bytes
PEQ2	702d:8500	702d:85FF	0001:8500	0001:85FF	256 Bytes
MBDRC2	702d:8600	702d:87FF	0001:8600	0001:87FF	512 Bytes
MVC	702d:A000	702d:Abff	0001:A000	0001:Abff	3 KB

Table 2: APE Address Map

Description	Address Start	Address End	Offset Start	Offset End	Default Length
MVC1	702d:A000	702d:A1ff	0001:A000	0001:A1ff	512 Bytes
MVC2	702d:A200	702d:A3ff	0001:A200	0001:A3ff	512 Bytes
MDMIF	702d:AC00	702d:B7FF	0001:AC00	0001:B7FF	3 KB
MDMIF0	702d:ac00	702d:adff	0001:AC00	0001:Adff	512 Bytes
MDMIF1	702d:ae00	702d:afff	0001:AE00	0001:Afff	512 Bytes
AHC	702d:B900	702d:BAFF	0001:B900	0001:BAFF	512 Bytes
MIXER	702d:BB00	702d:C2FF	0001:BB00	0001:C2FF	2 KB
MIXER1	702d:BB00	702d:C2FF	0001:BB00	0001:C2FF	2 KB
UART	702d:C300	702d:C4FF	0001:C300	0001:C4FF	512 Bytes
I2C	702d:C500	702d:C6FF	0001:C500	0001:C6FF	512 Bytes
GPIO	702d:C700	702d:C7FF	0001:C700	0001:C7FF	256 Bytes
FPGA_MISC	702d:C800	702d:C8FF	0001:C800	0001:C8FF	256Bytes
FPGA_CAR	702d:D000	702d:DFFF	0001:D000	0001:DFFF	4 KB
IQC1	702d:e000	702d:e1FF	0001:E000	0001:E1FF	512Bytes
IQC2	702d:e200	702d:e3FF	0001:E200	0001:D3FF	512Bytes
AHOST	702e:0000	702e:1FFF	0002:0000	0002:1FFF	8 KB
ADMA	702e:2000	702e:3FFF	0002:2000	0002:3FFF	8 KB
BSEA	702e:4000	702e:5FFF	0002:4000	0002:5FFF	8 KB
VCP	702e:6000	702e:7FFF	0002:6000	0002:7FFF	8 KB
AMISC	702e:C000	702e:DFFF	0002:C000	0002:DFFF	8 KB
AMISC_AMISC	702e:C000	702e:C7FF	0002:C000	0002:C7FF	2 KB
ADSP_ACTMON	702e:C800	702e:CBFF	0002:C800	0002:CBFF	1 KB
ABRIDGE	702e:E000	702e:E7FF	0002:E000	0002:E7FF	2 KB
AAS	702e:EF00	702e:EFFF	0002:EF00	0002:EFFF	256 B
AMC	702e:F000	702e:FFFF	0002:F000	0002:FFFF	4 KB
ACONNECT	702f:0000	702f:7FFF	0003:0000	0003:7FFF	32 KB
AGIC	702f:8000	702f:FFFF	0003:8000	0003:FFFF	32 KB

2.3 Available DRAM Address Ranges

Table 3 describes the various address regions that are available as DRAM in Tegra® X1 devices.

The address map available varies by master, because MMIO is not available to any direct MC client except the processors. These other direct MC clients are therefore able to access all DRAMs on 4GB systems, with the spaces reserved for MMIO on the processors available to them as DRAM. This may be used as a carve-out or for similar purposes.

Note: *MMIO in Table 3 below refers to the MMIO target, register, or non-DRAM region. Not all of the MMIO regions actually have an MMIO target.*

Physical/Virtual Address Range is available for Host1x clients for all apertures.

Table 3: DRAM Address Ranges

Aperture		Range	Size (MB)	Hardware Configuration Option	Physical DRAM Range				IOVA Range Available for:	
					2 GB	3GB	4 GB	>4 GB	BPMP-Lite	AHB Clients ?
DRAM		8000_0000 - FFFF_FFFF	2048	Always DRAM	DRAM	DRAM	DRAM	DRAM	Yes	Yes
DRAM2		1_0000_0000 - 2_7FFF_FFFF	6144	Always DRAM	N/A	N/A	N/A	DRAM	No	No
AHB_A2_rsvd		7E00_0000 - 7FFF_FFFF	32	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
AHB_A2		7C00_0000 - 7DFF_FFFF	32	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
AHB_A1_rsvd		7900_0000 - 7BFF_FFFF	48	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
AHB_A1		7800_0000 - 78FF_FFFF	16	Selectable	MMIO	MMIO	MMIO	MMIO	No	No
APB_rsvd		7400_0000 - 77FF_FFFF	64	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
APB		7000_0000 - 73FF_FFFF	64	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
ExtIO_rsvd		6900_0000 - 6FFF_FFFF	112	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
ExtIO		6800_0000 - 68FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
PPSB_rsvd		6100_0000 - 67FF_FFFF	112	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
PPSB		6000_0000 - 60FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
GART		5700_0000 - 5FFF_FFFF	144	Selectable (Note 1)	MMIO	MMIO	MMIO	MMIO	Yes	Yes
Graphics Host rsvd (HOST1X_DR_RSVD)		5500_0000 - 56FF_FFFF	32	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
Graphics Host (HOST1X_DR)		5400_0000 - 54FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
Host1x+PERIPH (Note 2)		5000_0000 - 50FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO		
Host1x+ PERIPH	Host1x	5000_0000 – 5003_FFFF	256KB	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
	PERIPHBASE	5004_0000 - 5005_FFFF	128KB	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
	MSELECT	5006_0000 - 5006_0FFF	4KB	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
RSVD		4B00_0000 - 4FFF_FFFF	80	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
RSVD		4900_0000 - 4AFF_FFFF	32	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
RSVD		4800_0000 - 48FF_FFFF	16	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
IRAM_rsvd		4100_0000 - 47FF_FFFF	112	Selectable	DRAM+	DRAM+	DRAM	DRAM+	Yes	Yes
IRAM		4000_0000 - 40FF_FFFF	16	Always MMIO	MMIO	MMIO	MMIO	MMIO	No	No
PCIE_A3		2000_0000 - 3FFF_FFFF	512	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes	Yes
PCIE_A2		0200_0000 - 1FFF_FFFF	480	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes	Yes

Table 3: DRAM Address Ranges

Aperture	Range	Size (MB)	Hardware Configuration Option	Physical DRAM Range				IOVA Range Available for:	
				2 GB	3GB	4 GB	>4 GB	BPMP-Lite	AHB Clients ?
PCIE_A1	0100_0000 - 01FF_FFFF	16	Selectable	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	DRAM/ MMIO	Yes	Yes
IROM_LOVEC	0000_0000 - 00FF_FFFF	16	Selectable (Note 3)	MMIO	MMIO	MMIO	MMIO	No	No

Notes:

1. The GART region, even though defined as configurable, needs to be programmed as MMIO because this region is actually the Kepler GPU MMIO space.
2. Host1x+PERIPH is split into multiple apertures.
3. The IROM_LOVEC region needs to be programmed as MMIO because this region is used to access the IROM and the exception vectors for both Cortex®-A57 CPU and ARM7AVP/BPMP-Lite and to ensure the CCPLEX and BPMP-Lite can access their exception vectors.
4. “DRAM” really means “Memory Controller”. In particular:
   - The GART range (5700_0000 – 5FFF_FFFF) always points to MMIO.
   - All configurable regions that do not have a real MMIO target (labeled as DRAM+) in the above table should be configured as DRAM to provide the largest and uniform IOVA view (in 2GB or 4GB system) for the BPMP-Lite and other clients that generate virtual addresses to access DRAM.
5. Requests from all clients (except the main CPU) are allowed to use SMMU translation. Any request address range that can reach the MC can be translated by the SMMU.
6. SMMU is a sub-unit in the memory controller. Address ranges marked as “No” in the table will not reach the memory controller.
7. If a region does not contain any DRAM (low 2GB region in a 34b system or system with only 2GB of physical memory), the various configurable regions still should be configured as DRAM to allow IOVA/DVA requests from BPMP to be passed through to the Memory Controller (MC).
8. For PCIe configuration and IOIO space, software needs to use SO (Strongly Ordered) or DEV (Device) mem type to guarantee 4 byte transactions.
9. For TZRAM access, software needs to mark the memory as WB cacheable and RW allocate to avoid partial byte enables.

Clients/Controllers are logically (static) grouped in different SWNAME/SWID. SMMU translation can be enabled/disabled for individual SWNAME. The following table gives the Client to Software Name mapping details.

Table 4: Client to Software Name Mapping

Software Group	Client	Description	SMMU Client Type
afi	csr_afir	PCIe reads	34 bits
	csw_afiw	PCIe writes	34 bits
ape	csr_aper	Audio Processing (APE) engine reads	32 bits
	csw_apew	Audio Processing (APE) engine writes	32 bits
avpc	csr_avpcarm7r	ARM7 BPMP-Lite reads via the avp_cache	AHBSLV
	csw_avpcarm7w	ARM7 BPMP-Lite writes via the avp_cache	AHBSLV
axiap	csr_axiapr	AXIAP reads	34 bits
	csw_axiapw	AXIAP writes	34 bits

Table 4: Client to Software Name Mapping

Software Group	Client	Description	SMMU Client Type
dc	csr_display0a	Display reads, window A	34bits
	csr_display0b	Display reads, window B	34 bits
	csr_display0c	Display reads, window C	34 bits
	csr_displayhc	Display reads, cursor	34 bits
	csr_displayt	Display reads, Window T	34 bits
	csr_displayd	Display reads, Window D	34 bits
dcb	csr_display0ab	Display reads, window A	34 bits
	csr_display0bb	Display reads, window B	34 bits
	csr_display0cb	Display reads, window C	34bits
	csr_displayhcb	Display reads, cursor	34 bits
etr	csr_etrr	ETR reads	34 bits
	csw_etrw	ETR writes	34 bits
gpu	csr_gpusrd	3D, ltcx reads	34 bits
	csw_gpuswr	3D, ltcx writes	34 bits
	csr_gpusrd2	3D, ltcx reads; instance 2 of GPU	34 bits
	csw_gpuswr2	3D, ltcx writes; instance 2 of GPU	34 bits
hc	csr_host1xdmar	Host channel data reads	32 bits
	csr_host1xr	Host indirect reads	32 bits
	csw_host1xw	Host indirect writes	32bits
hda	csr_hdar	High-definition audio (HDA) reads	34 bits
	csw_hdaw	High-definition audio (HDA) writes	34bits
isp2	csr_ispra	ISP Read client for Crossbar A	32 bits
	csw_ispwa	ISP Write client for Crossbar A	32 bits
	csw_ispwb	ISP Write client Crossbar B	32 bits
isp2b	csr_isprab	ISP Read client for Crossbar A; instance b of isp2	32 bits
	csw_ispwab	ISP Write client for Crossbar A; instance b of isp2	32 bits
	csw_ispwbb	ISP Write client Crossbar B; instance b of isp2	32 bits
mpcore	csr_mpcorer	Reads from Cortex-A57 4 CPU cores via the L2 cache	N/A[a]
	csw_mpcorew	Writes from Cortex-A57 4 CPU cores via the L2 cache	N/A[1]
nvdec	csr_nvdecsrd		32 bits
	csw_nvdecswr		32 bits
nvenc	csr_nvencsrd		32 bits
	csw_nvencswr		32 bits
nvjpg	csr_nvjpgsrd		32 bits
	csw_nvjpgswr		32 bits
ppcs	csr_ppcsahbdmar	AHB DMA engine reads	32 bits
	csr_ppcsahbslvr	AHB bus reads	32 bits
	csw_ppcsahbdmaw	AHB DMA engine writes	32 bits
	csw_ppcsahbslvw	AHB bus writes	32 bits
ptc	csr_ptcr	Misses from System Memory Management Unit (SMMU) Page Table Cache (PTC)	N/A[b]
sata	csr_satar	SATA reads	34 bits
	csw_sataw	SATA writes	34 bits

Table 4: Client to Software Name Mapping

Software Group	Client	Description	SMMU Client Type
sdmmc1a	csr_sdmmcra	SDMMCA memory read client	34 bits
	csw_sdmmcwa	SDMMCA memory write client	34 bits
sdmmc2a	csr_sdmmcraa	SDMMCB memory read client	34 bits
	csw_sdmmcwaa	SDMMCB memory write client	34 bits
sdmmc3a	csr_sdmmcr	SDMMC memory read client	34 bits
	csw_sdmmcw	SDMMC memory write client	34 bits
sdmmc4a	csr_sdmmcrab	SDMMCD memory read client	34 bits
	csw_sdmmcwab	SDMMCD memory write client	34 bits
se	csr_sesrd	SE Memory Return Data Client Description	AHBSLV
	csw_seswr	SE Memory Write Client Description	AHBSLV
tsec	csr_tsecsrd	TSEC Memory Return Data Client Description	32 bits
	csw_tsecswr	TSEC Memory Write Client Description	32 bits
tsecb	csr_tsecsrdb	TSEC Memory Return Data Client Description; instance B of TSEC	32 bits
	csw_tsecswrb	TSEC Memory Write Client Description; instance B of TSEC	32 bits
vi	csw_viw	VI Write client	32 bits
vic	csr_vicsrd	VIC reads	32 bits
	csw_vicswr	VIC writes	32 bits
xusb_dev	csr_xusb_devr	XUSB reads	34 bits
	csw_xusb_devw	XUSB_DEV writes	34 bits
xusb_host	csr_xusb_hostr	XUSB reads	AHBSLV
	csw_xusb_hostw	XUSB_HOST writes	AHBSLV

a. Cortex-A57 CPU accesses are translated by the MMU in the CPU not the SMMU.

b. SMMU misses do not use IOVA addresses.

The clients behind AHBSLV in the above table are:

- BPMP-Lite crossbar
- ARC controller
- AHB DMA controllers
- USB-OTG controller
- APB DMA controller
- Security Engine (SE)
- USB2 controllers

Each client in the above list has a SWID register field which controls whether the client maps to PPCS or PPCS1 SWNAME.

2.4 SMMU MC Client Address Ranges

All MC clients, except the SMMU and the Cortex-A57 cores, can be translated by the SMMU. The address ranges (either IOVA or PA addresses) that can be used by an SMMU MC client depend on the client. Each SMMU MC client can be assigned to one of three types: 34-bit clients, 32-bits clients, and AHBSLV clients. The table above shows the SMMU MC client type for all clients.

34-bit SMMU clients can send any address between 0_0000_0000 and 2_7FFF_FFFF to the MC.

32-bit SMMU clients can send any address between 0_0000_0000 and 0_FFFF_FFFF to the MC. Addresses above 0_FFFF_FFFF cannot be used by these clients.

AHBSLV clients can send any address between 0_0000_0000 and 0_FFFF_FFFF to the MC except those shown in the table below. Addresses above 0_FFFF_FFFF cannot be used by these clients.

Note: *If any of the selectable DRAM ranges shown in*

Selectable DRAM Address Ranges are not configured as DRAM per the recommendation, those address ranges are also not usable as MC addresses from AHBSLV clients.

Table 5: Unusable IOVA Address Ranges for AHBSLV Clients

Aperture	Range	Size (MB)
AHB_A2	7C00_0000 - 7DFF_FFFF	32
AHB_A1	7800_0000 - 78FF_FFFF	16
APB	7000_0000 - 73FF_FFFF	64
ExtIO	6800_0000 - 68FF_FFFF	16
PPSB	6000_0000 - 60FF_FFFF	16
Host1x	5000_0000 – 5003_FFFF	256
PERIPHBASE	5004_0000 - 5005_FFFF	128
MSELECT	5006_0000 - 5006_0FFF	4
Host1x+PERIPH	5000_0000 - 50FF_FFFF	16
IRAM	4000_0000 - 40FF_FFFF	16
IROM_LOVEC	0000_0000 - 00FF_FFFF	16

Table 6: Selectable DRAM Address Ranges

Aperture	Range	Size (MB)
AHB_A2_rsvd	7E00_0000 - 7FFF_FFFF	32
AHB_A1_rsvd	7900_0000 - 7BFF_FFFF	48
APB_rsvd	7400_0000 - 77FF_FFFF	64
ExtIO_rsvd	6900_0000 - 6FFF_FFFF	112
PPSB_rsvd	6100_0000 - 67FF_FFFF	112
Graphics Host rsvd (HOST1X_DR_RSVD)	5500_0000 - 56FF_FFFF	32
RSVD	4B00_0000 - 4FFF_FFFF	80
RSVD	4900_0000 - 4AFF_FFFF	32
RSVD	4800_0000 - 48FF_FFFF	16
IRAM_rsvd	4100_0000 - 47FF_FFFF	112

CHAPTER 3: INTERRUPT CONTROLLER

This section provides the mapping of the Tegra® X1 family processor interrupts. It also describes the architecture for routing and handling of these interrupts. It describes semaphores, which provide a mechanism for the processors to arbitrate for the use of various resources.

From the perspective of interrupts: devices (GPU, Memory Controller, Video encode/decode engines, and various I/O devices) are the sources of interrupts; and processors are the targets of interrupts. From sources, interrupts go to interrupt controllers which, based on configuration, prioritize and route them to the appropriate target processor. There are two different types of interrupt controllers used in Tegra X1 devices: the ARM® GIC-400 and the Legacy Interrupt Controller (LIC).

Glossary and Acronyms

Acronym	Description
BPMP-L	Boot and Power Management Processor - Lite
CPU	Unless specified otherwise, refers to Cortex®-A57 and/or Cortex-A53 CPUs
FIQ	Fast Interrupt Request
GIC-400	Also known as GIC. ARM Generic Interrupt Controller
IPI	Inter Processor Interrupt
IRQ	Interrupt Request
LIC	Legacy Interrupt Controller (used for the BPMP-L)
MSI	Message Signaled Interrupt
PPI	Private Peripheral Interrupts
SGI	Software Generated Interrupt
SMP	Symmetric Multiprocessors
SPI	Shared Peripheral Interrupts
WFE	Wait For Event (an ARM instruction)
WFI	Wait For Interrupt (an ARM instruction)

3.1 References

Some ARM documentation is required to fully understand the Tegra interrupt architecture. Refer to ARM's website for further details and to access these.

Document	Description
GIC-400 Architecture	ARM CoreLink GIC-400 Generic Interrupt Controller Technical Reference Manual (version r0p1)
Timers Architecture	ARM® Architecture Reference Manual ARMv8
Cortex-A57 TRM	ARM Cortex-A57 CPU complex Processor Technical Reference Manual

3.2 Interrupt Mapping

The following six tables show the mapping of the interrupts from system devices to the bit fields in the Tegra X1 interrupt controllers.

Table 7: Primary Interrupt Controller (PRI_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
31		31	SDMMC4	SDMMC	SDMMC4 Controller

Table 7: Primary Interrupt Controller (PRI_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
30		30	Unmapped		Unassigned
29	CPU	29	ARB_SEM_GNT_CPU	Semaphore	GNT.0 Arbitration Grant Status of CPU
28	BPMP-Lite	28	ARB_SEM_GNT_COP	Semaphore	GNT.1 Arbitration Grant Status of BPMP-Lite
27	CPU	27	Unmapped		Unassigned
26	CPU	26	APB_DMA_CPU	APB_DMA	APB Bridge DMA Controller (CPU)
25	BPMP-Lite	25	FC_INT	Flow Controller	Flow Controller to BPMP-Lite for Wake Sequence
24		24	PMC_INT	PMC	PMC Wake Events
23		23	SATA_CTL	SATA	SATA Controller Interrupt
22		22	Unmapped		Unassigned
21		21	USB2	USB	USB Device
20		20	USB	USB	USB Device
19		19	SDMMC3	SDMMC	SDMMC3 Controller
18		18	Unmapped		Unassigned
17		17	VII2C_INT	VII2C	I2C inside VI
16		16	VGPIO_INT	VGPIO	VGPIO Interrupt
15		15	SDMMC2	SDMMC	SDMMC2 Controller
14		14	SDMMC1	SDMMC	SDMMC1 Controller
13		13	SATA_RX_STAT	SATA	SATA RX Wake Up Interrupt
12		12	Unmapped		Unassigned
11		11	DPAUX_INT1	DPAUX_1	DPAUX Interrupt
10		10	QUAD_SPI	QUAD_SPI	Quad SPI Interrupt
9		9	NVDEC	NVDEC	NVDEC Interrupt
8		8	NVJPEG	NVJPEG	NVJPEG Interrupt
7	CPU	7	SHR_SEM_OUTBOX_ EMPTY	Semaphore	CPU Outbox Empty Interrupt
6	BPMP-Lite	6	SHR_SEM_OUTBOX_FULL	Semaphore	BPMP-Lite Outbox Full Interrupt
5	COPBPMP-Lite	5	SHR_SEM_INBOX_ EMPTY	Semaphore	BPMP-Lite Inbox Empty Interrupt
4	CPU	4	SHR_SEM_INBOX_FULL	Semaphore	CPU Inbox Full Interrupt
3		3	CEC	CEC	CEC General Interrupt
2		2	RTC	RTC	RTC Interrupt
1		1	TMR2	Timer	TMR2 Interrupt
0		0	TMR1	Timer	TMR1 Interrupt

Table 8: Secondary Interrupt Controller (SEC_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
63		31	I2C6	I2C	I2C6 Interrupt
62		30	CLDVFS	CL-DVFS	Close Loop DVFS control logic
61		29	Unmapped		Unassigned
60	COPBPMP-Lite	28	APB_DMA_COP	APB_DMA	APB-DMA Interrupt --BPMP-Lite
59		27	SPI1	SPI	SPI Controller (SBC1)

Table 8: Secondary Interrupt Controller (SEC_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
58		26	SE	SE	SE and TZRAM General Interrupt
57		25	USB3_DEV_PME	USB3	USB3_DEV_PME
56		24	USB3_DEV_SMI	USB3	USB3_DEV_SMI
55		23	GPIO5	GPIO (external)	GPIO4 Interrupt
54		22	Unmapped		Unassigned
53		21	I2C5	I2C	I2C5 Interrupt
52		20	Unmapped		Unassigned
51		19	EDP	SOC_THERM	EDP (Electrical Design Power) interrupt
50		18	TSEC	TSEC	Tegra HDCP Security Controller
49		17	XUSB_PADCTL	XUSB_PADCTL	Unassigned
48		16	THERMAL	SOC_THERM	SOC Thermal interrupt
47		15	Unmapped		Unassigned
46		14	UARTC	UART	UART-C
45		13	ACTMON	ActMon	ACTMON
44		12	USB3_DEV_HOST	USB3	USB3_DEV_HOST
43		11	USB3_HOST_PME	USB3	USB3_HOST_PME
42		10	TMR4	Timer	TMR4 Interrupt
41		9	TMR3	Timer	TMR3 Interrupt
40		8	USB3_HOST_SMI	USB3	USB3_HOST_SMI
39		7	USB3_HOST_INT	USB3	USB3_HOST_INT
38		6	I2C	I2C	I2C Interrupt
37		5	UARTB	UART	UART-B
36		4	UARTA	UART	UART-A
35		3	GPIO4	GPIO (external)	GPIO3 Interrupt
34		2	GPIO3	GPIO (external)	GPIO2 Interrupt
33		1	GPIO2	GPIO (external)	GPIO1 Interrupt
32		0	GPIO1	GPIO (external)	GPIO0 Interrupt

Table 9: Tertiary Interrupt Controller (TRI_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
95		31	Unmapped		Unassigned
94		30	Unmapped		Unassigned
93		29	SPI4	SPI	Serial Link 2 Sub-block Controller
92		28	I2C3	I2C	I2C3 Interrupt
91		27	Unmapped		Unassigned
90		26	UARTD	UART	UART-D
89		25	GPIO7	GPIO (external)	GPIO7 Interrupt
88		24	Unmapped		Unassigned
87		23	GPIO6	GPIO (external)	GPIO6 Interrupt
86		22	PMU_EXT	PMIC	External Power Management Chip Interrupt
85		21	Unmapped		Unassigned
84		20	I2C2	I2C	I2C2 Interrupt

Table 9: Tertiary Interrupt Controller (TRI_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
83		19	SPI3	SPI	SPI Controller (SBC3)
82		18	SPI2	SPI	SPI Controller (SBC2)
81		17	HDA	HDA	HDA Interrupt
80		16	TSECB	TSEC	TSECB Interrupt
79		15	Unmapped		Unassigned
78		14	EMC	EMC	EMC General Interrupt
77		13	MC	MC	MC General Interrupt
76		12	SOR	SOR (Display)	SOR General Interrupt
75		11	SOR1	SOR1	SOR1 General Interrupt
74		10	DISPLAYB	DISPLAYB	Display B General Interrupt
73		9	DISPLAY	DISPLAY	Display A General Interrupt
72		8	VIC	VIC	Video Image Compositer General Interrupt
71		7	ISP	ISP	ISP General Interrupt
70		6	ISPB	ISPB	ISPB General Interrupt
69		5	VI	VI	VI General Interrupt
68		4	NVENC	NVENC	NVENC General Interrupt
67	CPU	3	HOST1X_GEN_CPU	HOST1X	CPU General Interrupt
66	COP1	2	HOST1X_GEN_COP	HOST1X	BPMP-Lite General Interrupt
65	CPU	1	HOST1X_SYNCPT_CPU	HOST1X	CPU SyncPt Interrupt
64	COP1	0	HOST1X_SYNCPT_COP	HOST1X	BPMP-Lite SyncPt Interrupt

Table 10: Quaternary Interrupt Controller (QUAD_CTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
127		31	Unmapped		Unassigned
126		30	CAR	CAR	CAR Interrupt
125		29	GPIO8	GPIO (external)	GPIO8 Interrupt
124		28	WDT_AVP	Timer	Watchdog BPMP-Lite Interrupt
123		27	WDT_CPU	Timer	Watchdog CPU Interrupt
122		26	Unmapped		Unassigned
121		25	TMR5	Timer	TMR5 Interrupt
120		24	I2C4	I2C	I2C4 Interrupt
119		23	APB_DMA_CH15	APB_DMA	APB-DMA Channel 15
118		22	APB_DMA_CH14	APB_DMA	APB-DMA Channel 14
117		21	APB_DMA_CH13	APB_DMA	APB-DMA Channel 13
116		20	APB_DMA_CH12	APB_DMA	APB-DMA Channel 12
115		19	APB_DMA_CH11	APB_DMA	APB-DMA Channel 11
114		18	APB_DMA_CH10	APB_DMA	APB-DMA Channel 10
113		17	APB_DMA_CH9	APB_DMA	APB-DMA Channel 9
112		16	APB_DMA_CH8	APB_DMA	APB-DMA Channel 8
111		15	APB_DMA_CH7	APB_DMA	APB-DMA Channel 7
110		14	APB_DMA_CH6	APB_DMA	APB-DMA Channel 6
109		13	APB_DMA_CH5	APB_DMA	APB-DMA Channel 5

Table 10: Quaternary Interrupt Controller (QUAD_CTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
108		12	APB_DMA_CH4	APB_DMA	APB-DMA Channel 4
107		11	APB_DMA_CH3	APB_DMA	APB-DMA Channel 3
106		10	APB_DMA_CH2	APB_DMA	APB-DMA Channel 2
105		9	APB_DMA_CH1	APB_DMA	APB-DMA Channel 1
104		8	APB_DMA_CH0	APB_DMA	APB-DMA Channel 0
103		7	APE_INT0	APE	APE
102		6	APE_INT1	APE	APE
101		5	AVP_CACHE	ARM7 Cache	ARM7 Cache Interrupt
100		4	Unmapped		Unassigned
99		3	PCIE_MSI	PCIe®	PCIe message signaled interrupt
98		2	PCIE_INT	PCIe	PCIe catch-all error/legacy interrupts
97		1	Unmapped		Unassigned
96		0	DTV	DTV	DTV

Table 11: Fifth Interrupt Controller (PENTA_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
159		31	DPAUX	DPAUX	DPAUX General Interrupt
158		30	GPU_NONSTALL	GPU	GPU Interrupt
157		29	GPU_STALL	GPU	GPU Interrupt
156		28	TMR0	Timer	TMR0 Interrupt
155		27	TMR9	Timer	TMR9 Interrupt
154		26	TMR8	Timer	TMR8 Interrupt
153		25	TMR7	Timer	TMR7 Interrupt
152		24	TMR6	Timer	TMR6 Interrupt
151		23	SDMMC4_SYS	SDMMC	SDMMC4 interrupt for standard driver (e.g., WOA)
150		22	SDMMC3_SYS	SDMMC	SDMMC3 interrupt for standard driver (e.g., WOA)
149		21	SDMMC2_SYS	SDMMC	SDMMC2 interrupt for standard driver (e.g., WOA)
148		20	SDMMC1_SYS	SDMMC	SDMMC1 interrupt for standard driver (e.g., WOA)
147		19	CPU3_PMU_INTR	A15 CPU3	CPU3 Performance Management Unit (PMU) Interrupt
146		18	CPU2_PMU_INTR	A15 CPU2	CPU2 PMU Interrupt
145		17	CPU1_PMU_INTR	A15 CPU1	CPU1 PMU Interrupt
144		16	CPU0_PMU_INTR	A15 CPU0	CPU0 PMU Interrupt
143		15	APB_DMA_CH31	APB_DMA	APB-DMA Channel 31
142		14	APB_DMA_CH30	APB_DMA	APB-DMA Channel 30
141		13	APB_DMA_CH29	APB_DMA	APB-DMA Channel 29
140		12	APB_DMA_CH28	APB_DMA	APB-DMA Channel 28
139		11	APB_DMA_CH27	APB_DMA	APB-DMA Channel 27
138		10	APB_DMA_CH26	APB_DMA	APB-DMA Channel 26
137		9	APB_DMA_CH25	APB_DMA	APB-DMA Channel 25
136		8	APB_DMA_CH24	APB_DMA	APB-DMA Channel 24

Table 11: Fifth Interrupt Controller (PENTA_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
135		7	APB_DMA_CH23	APB_DMA	APB-DMA Channel 23
134		6	APB_DMA_CH22	APB_DMA	APB-DMA Channel 22
133		5	APB_DMA_CH21	APB_DMA	APB-DMA Channel 21
132		4	APB_DMA_CH20	APB_DMA	APB-DMA Channel 20
131		3	APB_DMA_CH19	APB_DMA	APB-DMA Channel 19
130		2	APB_DMA_CH18	APB_DMA	APB-DMA Channel 18
129		1	APB_DMA_CH17	APB_DMA	APB-DMA Channel 17
128		0	APB_DMA_CH16	APB_DMA	APB-DMA Channel 16

Table 12: Sixth Interrupt Controller (HEXA_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
191		31	Unmapped		Unassigned
190		30	Unmapped		Unassigned
189		29	Unmapped		Unassigned
188		28	Unmapped		Unassigned
187		27	Unmapped		Unassigned
186		26	Unmapped		Unassigned
185		25	Unmapped		Unassigned
184		24	Unmapped		Unassigned
183		23	Unmapped		Unassigned
182		22	Unmapped		Unassigned
181		21	Unmapped		Unassigned
180		20	Unmapped		Unassigned
179		19	TMR13	Timer	TMR13 Interrupt
178		18	TMR12	Timer	TMR12 Interrupt
177		17	TMR11	Timer	TMR11 Interrupt
176		16	TMR10	Timer	TMR10 Interrupt
175	CPU	15	MSELECT_ERROR	Mselect	Mselect Error Interrupt
174		14	MPCORE_CTIIRQ3	CPU	CPU3 CTI interrupt
173		13	MPCORE_CTIIRQ2	CPU	CPU2 CTI interrupt
172		12	MPCORE_CTIIRQ1	CPU	CPU1 CTI interrupt
171		11	MPCORE_CTIIRQ0	CPU	CPU0 CTI interrupt
170		10	TMR_SHARED	Timer	TMR_SHARED interrupt
169		9	FLOW_RSM_COP	FlowCtlr	FLOW - RSM1 Resume for BPMP-Lite
168		8	FLOW_RSM_CPU	FlowCtlr	FLOW - RSM0 Resume for CPU
167		7	Unmapped		Unassigned
166		6	Unmapped		Unassigned
165		5	Unmapped		Unassigned
164		4	EVENT_GPIO_C	GPIO (External)	Event Detector GPIO Port C
163		3	EVENT_GPIO_B	GPIO (External)	Event Detector GPIO Port B
162		2	EVENT_GPIO_A	GPIO (External)	Event Detector GPIO Port A

Table 12: Sixth Interrupt Controller (HEXA_ICTLR) Mapping

Global Interrupt Number	Interrupt Target	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description
161		1	MPCORE_ INTERRIRQ	CCPLEX	CPU INTERRIRQ
160		0	MPCORE_ AXIERRIRQ	CCPLEX	CPU AXIERRIRQ

3.3 Functional Description

3.3.1 Interrupt Handling Mechanism

The two different types of interrupt controllers (GIC-400 and LIC) receive interrupts from devices (i.e., hardware interrupts also known as SPIs) or processors (i.e., software interrupts including IPI), arbitrate them, and send them to the appropriate target processor(s). From an interrupt perspective, there are two different types of processors: CPUs and BPMP (ARM7 BPMP-Lite). The GIC-400 is the interrupt controller for the CPUs, and the LIC is the interrupt controller for the ARM7 BPMP-Lite. Any processor can initiate a software interrupt, targeted to any one or more processors (including itself). However, IPIs can only be initiated by a Cortex-A57 CPU to any one or more Cortex-A57 CPUs (including itself).

There are 192 hardware interrupts in Tegra X1 devices. Interrupt sources are allocated one or more interrupts as required. The 192 interrupts are grouped into slices of 32, where each slice can be configured independently.

3.3.1.1 ARM Processors' IRQ and FIQ

ARM processors (Cortex-A57, and ARM7) each have two input pins to receive two different types of interrupts. These interrupts are called IRQ (Interrupt Request) and FIQ (Fast Interrupt Request). The interrupts are implemented as active-low pins on the ARM processor input; thus the processor pins are named nIRQ and nFIQ.

The ARM processor goes into the IRQ mode or the FIQ mode depending up on which interrupt is activated. Generally, interrupts that require low latency or are time critical are configured as FIQ. All other interrupts are configured as IRQ. Non-secure interrupts can only be IRQ.

3.3.1.2 Interrupt Controllers

The interrupt controller receives interrupts from a large number of sources. The interrupt sources can be assigned a target processor, a type (IRQ versus FIQ), priority levels, etc., by configuring interrupt-controller registers. The interrupt controller arbitrates among different incoming interrupts and sends the interrupt to the nIRQ or nFIQ pin of the targeted processor.

In general, any incoming hardware interrupt can be routed to either nIRQ or nFIQ pin of any of the processors.

The Legacy Interrupt Controller (LIC) is primarily used for BPMP (ARM7). But it is also used for generating interrupts as wake events for CPUs. This is an important use case when the core is in retention. All of the device hardware interrupt signals are sent to the LIC first, which routes them to the ARM7 BPMP-Lite as well as forwards them to the GIC. The LIC also provides a software set/clear mechanism for all of the interrupts.

The interrupt controller used for CPUs (Cortex-A57 CPUs) is called GIC-400. Because the internal interrupt controller/interface incorporated in the Cortex-A57 CPU does not include a distributor function, the Cortex-A57 CPU is configured to rely on an external interrupt controller. The Tegra X1 ARM CCPLEX uses a GIC-400, an ARM IP, as an external interrupt controller. The GIC implements GIC Security Extensions and GIC Virtualization Extensions. The GIC is configured to support 4 interfaces

3.3.1.3 GIC Interrupt Sources

The vGIC supports 192 SPI interrupts and 32 PPI and SGI interrupts. All interrupts are identified by a unique ID. There are three types of interrupt sources for the GIC and vGIC: SGIs, PPIs, and SPIs.

Software Generated Interrupts (SGIs)

SGIs are software interrupts which are generated by writing to the Software Generated Interrupt register (GICD_SGIR). Each CPU interface can generate a maximum of 16 SGIs, with ID0-15 for each target processor. The SGIs are also referred to as IPIs (Inter Processor Interrupts).

Private Peripheral Interrupts (PPIs)

PPIs are interrupts generated by a peripheral that is specific to a single processor.

There are seven PPIs for each Cortex-A57 processor: virtual maintenance interrupt (ID25), hypervisor timer interrupt (ID26), virtual timer interrupt (ID27), legacy nFIQ (ID28), secure physical timer interrupt (ID29), non-secure physical timer interrupt (ID30), and legacy nIRQ (ID31).

Tegra X1 devices use all PPIs.

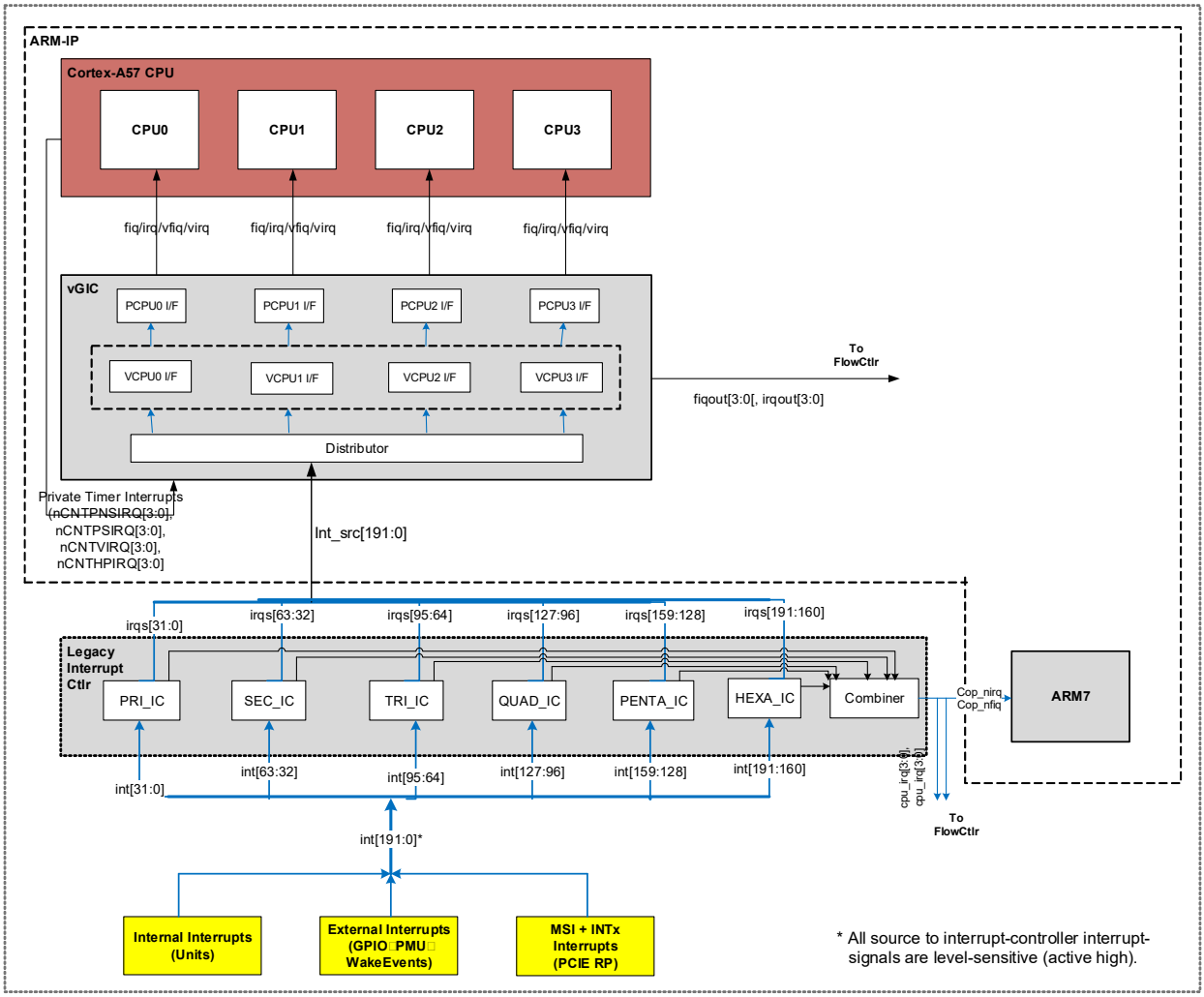
Shared Peripheral Interrupts (SPIs)

SPIs are external hardware interrupts which are generated by asserting signals on GIC or vGIC input pins (called IRQs). From a Tegra hardware perspective, these are the interrupts which are generated by various units to be sent to interrupt controllers. Tegra X1 devices use 192 of the possible 480 SPIs generated through the IRQS[191:0] pins. SPIs start at ID32. Tegra X1 devices use level-sensitive SPIs only.

3.3.1.4 Interrupt Routing

Tegra hardware interrupts (also known as shared peripheral interrupts) can be generated by internal SoC units, external events (through GPIO, USB, etc.), or PCIe® unit (MSI or legacy INTx, or Wake interrupts). All of these hardware interrupts are routed through sideband active-high level-sensitive signals to the LIC. The MSI and INTx (from external PCIe devices) are intercepted by the PCIe unit and converted to a sideband interrupt signal. The PCIe unit ensures MSI ordering; that is, it ensures that upstream writes ahead of MSI have been completed/acknowledged before asserting an interrupt signal to the LIC. Also, the PCIe unit captures MSI information into a register that software can read to identify MSI vectors, etc.

Figure 3: Interrupt Routing (Top-Level View)



As shown in the figure above, hardware interrupts are routed to the interrupt controllers (called IC32) within the LIC, where each IC32 handles 32 interrupts. The interrupt signals are synchronized within the LIC to the system clock. The synchronized signals are combined with software set/clear bits per interrupt. This allows software to set or clear individual interrupts, provided the corresponding interrupt is not asserted by hardware. The resulting interrupt signals are broadcast to the GIC-400 for further processing.

Within the LIC, the interrupt signals after combining with Software Set/Clear bits are also used by flow controller as Wake Events (after masking) and by BPMP-Lite as Interrupts (after masking).

3.3.1.5 Interrupt Blocking to Support Retention

The Tegra X1 device implements blocking of interrupts routed to the GIC, which supports the CPU retention state. The block implementing this feature is represented in Figure 4 as Blocking.

To support retention, the LIC contains a one shot disable for all the interrupts. When the system goes into retention, BPMP software sets this bit to disable the interrupts.

The Flow Controller watches all the interrupts triggered and triggers the BPMP to bring the core out of retention when any interrupt bit is asserted. Once the system is out of retention, the BLOCK _CCPLEX_GIC_INTR bit is cleared, and the interrupt is serviced by the GIC.

Refer to Chapter 18: Flow Controller for more information on retention.

3.3.2 Generic Interrupt Controller (GIC-400)

This section describes the Cortex-A57 GIC (also known as GIC-400). The GIC-400 is an implementation of the ARM® GICv2 architecture. The Tegra X1 processor does not use the built-in GIC interface logic. For more information, refer to the ARM Cortex-A57 MPCore Processor Technical Reference Manual for the GIC implementation specific specification and the CoreLink™ GIC-400 Generic Interrupt Controller Technical Reference Manual.

The GIC-400 consists of the physical CPU interface. In addition, to support CPU virtualization it consists of the virtual CPU Interface which is configured by the Hypervisor and used by the virtual machine (VM).

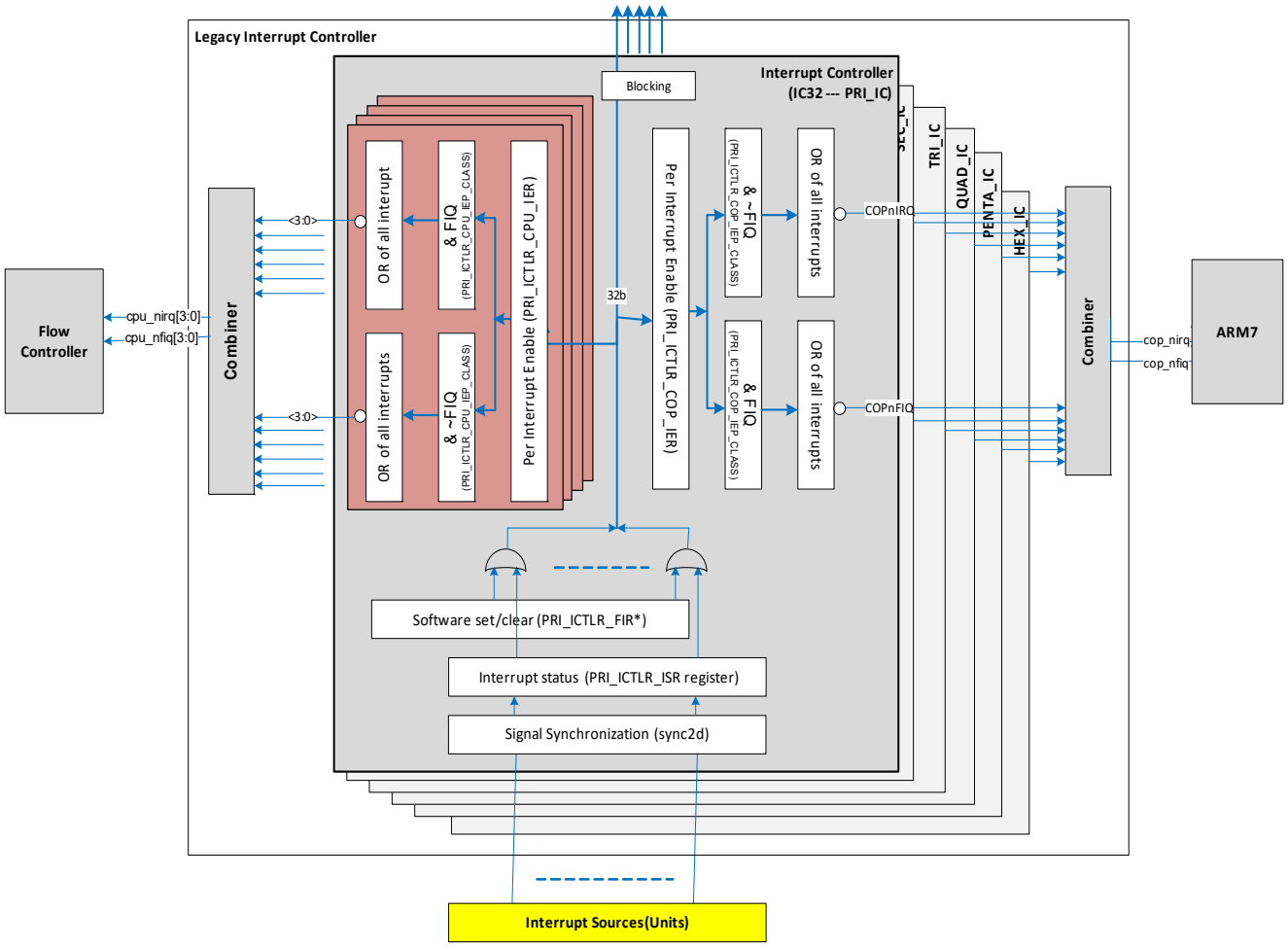
The GIC-400 presents the Flow Controller with four sets of FIQOUT and IRQOUT signals, one set for each CPU core. These signals serve as an indication (to the Flow Controller) that there are outstanding interrupts targeting one or more CPU cores. When one or more CPU cores are in a power state deeper than C1 (i.e., STANDBYWFx) such as retention or power gated state, upon observing an interrupt, the Flow Controller will initiate a power sequence to bring the respective CPU core or CPU cluster back into C0 (or CC0, Active) state.

The GIC-400 interrupt controller resides on the VDD_SoC power rail. The GIC-400 will not be externally power gated or clock gated. As long as the VDD_SoC rail is powered on, the GIC-400 will be powered on as well. With the exception of the LP0 power state, during which GIC-400 is powered down, the GIC-400 will be monitoring the interrupt requests throughout all the CCPLEX and SoC power states and route them to the Flow Controller, indicating that a pending interrupt to the CPU cores. In retention as explained above the cores would be idle and all interrupts to the GIC would be blocked. So, the GIC will not generate any interrupt signals.

3.3.3 Legacy Interrupt Controller (LIC)

Interrupts from various Tegra X1 units are routed to the LIC. The LIC consists of six interrupt controllers (called IC32); each IC32 handles 32 interrupts (for a total of 192 interrupts). Each IC32 generates five sets of IRQ/FIQ interrupts: one for BPMP (ARM7 BPMP-Lite) and the other ones for CPU<3:0>. The interrupts for CPU<3:0> are used by the flow controller only for wake events. A use case for this is explained in Section 3.3.1.5: Interrupt Blocking to Support Retention.

Figure 4: Legacy Interrupt Controller Block Diagram



The following text discusses interrupts as they relate to the BPMP-Lite IRQ/FIQ interrupts. Unless specified otherwise, this also applies to the CPU<3:0> IRQ/FIQ interrupts.

This is a general LIC functional description that assumes blocking is not enabled. See Section 3.3.1.5: Interrupt Blocking to Support Retention for information on blocking.

3.3.3.1 LIC Functional Description

Each IC32 in the LIC handles 32 interrupts. As shown in above figure, the IC32 synchronizes the incoming hardware interrupt requests and combines (bitwise OR) them with software interrupts (FIR). The software interrupt can set be set or cleared by the FIR register. The combined software/hardware interrupt signals are broadcast to the GIC-400, and are used by the IC32 for targeting interrupts to the BPMP-Lite. This allows any interrupt to be routed to any one or more processors (Cortex-A57/ ARM7) in a Tegra system. Note that routing can be changed dynamically. In addition, any of these interrupts can be mapped to either IRQ or FIQ.

The interrupt routing (to the BPMP-Lite) is achieved by configuring the Interrupt Enable Register (IER) and the Interrupt Class Registers (IEP_CLASS). Each IC32 has a set of IER and IEP Class registers. The BPMP-Lite registers are called COP_IER and COP_IEP_CLASS. Similarly, the CPU IER/IEP Class Registers are called CPU_IER/CPU<1,2,3>_IER and CPU_IEP_CLASS/CPU_IEP_CLASS.

When a 1 is set in the proper bit position in the COP_IER register, that particular source is capable of interrupting the BPMP-Lite. Also, the class is set in the proper bit position of COP_IEP_CLASS for the corresponding interrupt to be routed as IRQ or FIQ.

The interrupt status register (ISR) allows the processor to view the state of the pending hardware interrupt requests regardless of the bit enables programmed in COP_IER. The forced interrupt status register (FIR) allows the software to selectively force set or clear specific interrupts. The read-only VIRQ/VFIQ allows the BPMP-Lite to determine the source of the interrupt request(s) causing the processor to enter the interrupt service routine.

The nIRQ signals generated by six IC32 controllers are logically ANDed in the combiner to generate the final nIRQ which is routed to the BPMP-Lite. Similarly, nFIQ signals are logically ANDed in the combiner to generate nFIQ which is routed to the BPMP-Lite.

3.3.3.2 LIC Registers Description

Each IC32 has sixteen 32-bit registers. The BPMP-Lite and CPU<n> interrupt enable registers (COP_IER/CPU_IER) indicate the interrupts enabled for the BPMP-Lite and CPU<3:0>, respectively. These registers have their respective SET and CLR registers that allow setting or clearing individual bits of these registers without the need of a Read-Modify-Write cycle.

The BPMP-Lite/CPU Interrupt Class Register (COP_IEP_CLASS/CPU_IEP_CLASS) controls whether the interrupt will be routed to the IRQ or the FIQ interrupt of the respective processor.

The Forced Interrupt Status Register (FIR) allows the software to force the set and clear of individual interrupts. This can be used for debugging/testing purposes or can be used in the working system. Note that software cannot force set/clear an interrupt which is already asserted by a hardware source. The FIR has its corresponding SET and CLR registers.

The Valid Interrupt Status Register (VIRQ_COP/VIRQ_CPU) and Valid FIQ Interrupt Status Register (VFIQ_COP/VFIQ_CPU) indicate the currently active interrupts that are valid on the respective pins (nIRQ or nFIQ).

The differences between ISR and VFIQ/VIRQ registers are:

- The VFIQ/VIRQ registers indicate the interrupt which is sent to the processor (i.e., these registers indicate interrupts outgoing from the interrupt controller). On the other hand, the ISR indicates hardware interrupts incoming into the interrupt controller.
  - The VFIQ/VIRQ registers indicate the interrupts set by hardware or by software. The ISR register indicates registers set by only hardware.
  - The VFIQ/VIRQ registers only show the enabled interrupts. ISR shows the interrupt status irrespective of whether the interrupt is enabled or not.
  - The ISR shows the interrupt to be active whether it is routed to the FIQ pin or to the IRQ pin. The VFIQ/VIRQ registers contain only the routed interrupts.

The VIRQ_COP and VIRQ_CPU registers are calculated as follows.

VIRQ_COP = (ISR **|** FIR) **&** IER_COP **&** (**~**COP_IEP_CLASS)

VIRQ_CPU<n> = (ISR **|** FIR) **&** IER_CPU<n> **&** (**~**CPU_IEP_CLASS)

Similarly, the VFIQ_COP and VFIQ_CPU<n> registers are generated as follows.

VIRQ_COP = (ISR **|** FIR) **&** IER_COP **&** COP_IEP_CLASS

VIRQ_CPU<n> = (ISR **|** FIR) **&** IER_CPU<n> **&** CPU<n>_IEP_CLASS

3.3.4 Routing of Interrupt Signals from the LIC to the Flow Controller

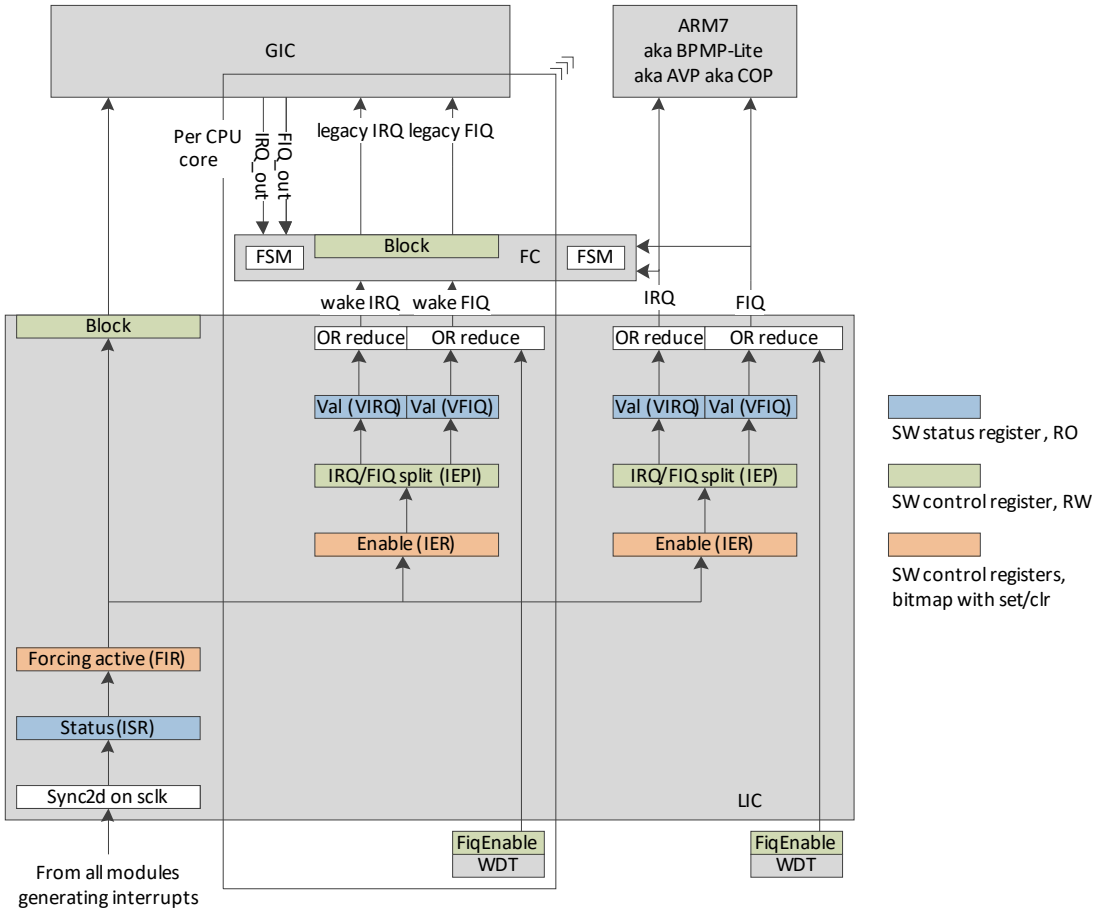
The interrupt signals are routed to the GIC-400 without any masking.

The interrupt signals are routed to the Flow Controller after masking with the Interrupt Enable Registers. Wake IRQ/FIQ at the LIC to FC interface in Figure 5 are the same as the signals CPU_nIRQ/CPU_nFIQ[3:0] in Figure 3.

The VFIQ signal generated for each core is combined with the Watch Dog Timer. Refer to Chapter 8: Timers for more information on the Watch Dog Timer.

The flow controller routes the interrupt signals to the legacy FIQ/ IRQ signal present for each core in the GIC. The legacy FIQ is used by the FIQ debugger.

Figure 5: Signal Routing with Respect to the LIC



3.3.5 Arbitration Semaphores

Arbitration semaphores provide a mechanism by which the CPU and BPMP-Lite can arbitrate for the use of various resources. These semaphores provide a hardware locking mechanism, so that when a processor is already using a resource, the second processor is not granted that resource. There are 32 bits of Arbitration semaphores provided in the system. The hardware does not enforce any resource association to these bits. It is left to the software to assign and use these bits. Any processor that needs to access a particular resource will request for the corresponding bit in the Arbitration semaphores by writing a one to that bit in the Arbitration Semaphore Request register (SMP_GET register). Software will then check the corresponding bit in the Semaphore Granted Status register (SMP_GNT_ST register)

If the requesting processor has been granted the resource, then the status returned will be a one. Alternately, the processor can configure the interrupt controller to generate an interrupt when the resource becomes available.

When the processor has finished using the resource, it releases the resource by writing a one to the corresponding bit in the Arbitration Semaphore Put Request register (SMP_PUT register). Additionally, pending request status is provided through the Arbitration Request Pending Status register (SMP_REQ_ST register).

The Primary Interrupt Controller (PRI_ICTLR) supports Arbitration Grant Interrupts. When a processor is granted access to a resource, the Arbitration semaphore module can be programmed to send a Grant signal to the interrupt controller. The interrupt controller can then interrupt the processor to indicate that it has been granted exclusive access to a particular resource. The individual ARB_GNT bits are ORed together and presented to the Primary Interrupt controller as Interrupt Number 29.

The Arbitration/Grant interrupts mechanism registers are described in the Arbitration Chapter 4: Semaphores of this document.

3.3.6 Interrupts Used for Processor Power-Ups

The interrupt signals (nIRQ and nFIQ per processor) are routed to the Flow Controller unit from which they could also continue to either processor as allowed by the flow controller. Based on configuration, the flow controller uses them to power up the corresponding CPU. Refer to Chapter 18: Flow Controller in this TRM for the details of processor wake-up.

3.4 Interrupt Registers

3.4.1 Interrupt Controller Registers

> ***Note:*** *The interrupt controller (IC32) register descriptions do not follow the standard register table protocol described in Section 1.4: Reading Register Tables in the Introduction chapter. Each register's relative offset, access type, and power-on reset value are located in Table 14. The bit descriptions for each register are defined in Table 15 according to interrupt controller (IC32).*

Each IC32 has a set of 34 registers with names using the form <ID>_<FUNCTION>_0:

- ID identifies the specific IC32: PRI_ICTLR, SEC_ICTLR, TRI_ICTLR, QUAD_ICTLR, PENTA_ICTLR, HEXA_ICTLR. Based on the IC32, ID defines a base address and mapping (see below).
- FUNCTION identifies the use for this register. Refer to the Section 3.3.3.2: LIC Registers Description for more details. FUNCTION is defined as one of the following:
  - VIRQ_<DEST> and VFIQ_<DEST>: indicates the Valid status, as an IRQ or FIQ, respectively, for this destination. DEST is BPMP-Lite, CPU, CPU1, CPU2, or CPU3.
  - ISR: indicates the latched Interrupt Status
  - FIR: indicates Force Interrupt
  - FIR_SET and FIR_CLR: to control FIR, bits set to 1 are set or cleared, respectively, in FIR.
  - <DEST>_IER: indicates that the corresponding event is enabled as an interrupt
  - <DEST>_IER_SET and <DEST>_IER_CLR: to control IER, bits equal to 1 are set or cleared, respectively, in IER.
  - <DEST>_IEP_CLASS: Interrupt Enable Priority Class, a bit set to 0 indicates IRQ, and set to 1 indicates FIQ.

Table 13 associates the following with each IC32: the ID (used in the register names), the base offset (address map of the five IC32s), and interrupt mapping reference.

Table 13: IC32 Characteristics

ID	Base Offset	Mapping
PRI_ICTLR	0x000	See Table 7: Primary Interrupt Controller (PRI_ICTLR) Mapping
SEC_ICTLR	0x100	See Table 8: Secondary Interrupt Controller (SEC_ICTLR) Mapping
TRI_ICTLR	0x200	See Table 9: Tertiary Interrupt Controller (TRI_ICTLR) Mapping
QUAD_ICTLR	0x300	See Table 10: Quaternary Interrupt Controller (QUAD_CTLR) Mapping
PENTA_ICTLR	0x400	See Table 11: Fifth Interrupt Controller (PENTA_ICTLR) Mapping
HEXA_ICTLR	0x500	See Table 12: Sixth Interrupt Controller (HEXA_ICTLR) Mapping

The table below summarizes all 34 registers for each IC32 (where <ID> is the controller name), their relative offsets, register access types (RO, WO, or RW), and their 32-bit power-on reset values. The actual offset for each register is the base offset (from the table above) plus the relative offset. The PRI_ICTLR, SEC_ICTLR, TRI_ICTLR, QUAD_ICTLR, PENTA_ICTLR, and HEXA_ICTLR have the same set of registers at the same address offset with the exception of the ARBGNT registers which are only in PRI_ICTLR. The ARBGNT registers are defined in Section 3.4.2: Arb_gnt Specific Interrupt Controller Registers.

Table 14: Interrupt Controller IC32 Register Set

Name	Relative Offset	Read/ Write	Reset	Remark
<ID>_VIRQ_CPU_0	0x00	RO	x	Valid Interrupt Request Status for CPU0 Register
<ID>_VIRQ_COP_0	0x04	RO	x	Valid Interrupt Request Status for BPMP-Lite Register
<ID>_VFIQ_CPU_0	0x08	RO	x	FIQ Valid Interrupt Status for CPU0 Register
<ID>_VFIQ_COP_0	0x0C	RO	x	FIQ Valid Interrupt Status for BPMP-Lite Register
<ID>_ISR_0	0x10	RO	x	Latched Interrupt Status Register
<ID>_FIR_0	0x14	RO	x	Forced Interrupt Status Register
<ID>_FIR_SET_0	0x18	WO	0	Force Interrupt Register Set Register
<ID>_FIR_CLR_0	0x1C	WO	0	Force Interrupt Register Clear Register
<ID>_CPU_IER_0	0x20	RO	x	Enabled Interrupt Source for CPU0 Register (0=disabled)
<ID>_CPU_IER_SET_0	0x24	WO	0	Set Interrupt Enable for CPU0 Register
<ID>_CPU_IER_CLR_0	0x28	WO	0	Clear Interrupt Enable for CPU Register
<ID>_CPU_IEP_CLASS_0	0x2C	RW	0	CPU0 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)
<ID>_COP_IER_0	0x30	RO	x	Enabled Interrupt Source for BPMP-Lite Register (0=disabled
<ID>_COP_IER_SET_0	0x34	WO	0	Set Interrupt Enable for BPMP-Lite Register
<ID>_COP_IER_CLR_0	0x38	WO	0	Clear Interrupt Enable for BPMP-Lite Register
<ID>_COP_IEP_CLASS_0	0x3C	RW	0	BPMP-Lite Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)
<ID>_VIRQ_CPU1_0	0x60	RO	x	Valid Interrupt Request Status for CPU1 Register
<ID>_VFIQ_CPU1_0	0x64	RO	x	FIQ Valid Interrupt Status for CPU Register
<ID>_CPU1_IER_0	0x68	RO	x	Enabled Interrupt Source for CPU1 Register (0=disabled)
<ID>_CPU1_IER_SET_0	0x6C	WO	0	Set Interrupt Enable for CPU1 Register
<ID>_CPU1_IER_CLR_0	0x70	WO	0	Clear Interrupt Enable for CPU1 Register
<ID>_CPU1_IEP_CLASS_0	0x74	RW	0	CPU1 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)
<ID>_VIRQ_CPU2_0	0x78	RO	x	Valid Interrupt Request Status for CPU2 Register
<ID>_VFIQ_CPU2_0	0x7C	RO	x	FIQ Valid Interrupt Status for CPU2 Register
<ID>_CPU2_IER_0	0x80	RO	x	Enabled Interrupt Source for CPU2 Register (0=disabled)
<ID>_CPU2_IER_SET_0	0x84	WO	0	Set Interrupt Enable for CPU2 Register
<ID>_CPU2_IER_CLR_0	0x88	WO	0	Clear Interrupt Enable for CPU2 Register
<ID>_CPU2_IEP_CLASS_0	0x8c	RW	0	CPU2 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)
<ID>_VIRQ_ CPU3_0	0x90	RO	x	Valid Interrupt Request Status for CPU3 Register
<ID>_VFIQ_ CPU3_0	0x94	RO	x	FIQ Valid Interrupt Status for CPU3 Register
<ID>_CPU3_IER_0	0x98	RO	x	Enabled Interrupt Source for CPU3 Register (0=disabled)
<ID>_CPU3_IER_SET_0	0x9C	WO	0	Set Interrupt Enable for CPU3 Register
<ID>_CPU3_IER_CLR_0	0xA0	WO	0	Clear Interrupt Enable for CPU3 Register
<ID>_CPU3_IEP_CLASS_0	0xA4	RW	0	CPU3 Interrupt Enable Priority Class Register (1=FIQ/0=IRQ)

The following table defines the bits within each of the five IC32 interrupt controller's registers. To calculate the global interrupt number of each bit, add the bit location (0 to 31) to the base interrupt value for the specific interrupt controller.

Table 15: IC32 Register Bit Descriptions

Bit	First IC32 (PRI_ICTLR) Bits (Base Interrupt = 0)	Second IC32 (SEC_ICTLR) Bits (Base Interrupt = 32)	Third IC32 (TRI_ICTLR) Bits (Base Interrupt = 64)	Fourth IC32 (QUAD_ICTLR) Bits (Base Interrupt = 96)	Fifth IC32 (PENTA_ICTLR) Bits (Base Interrupt = 128)	Sixth IC32 (HEXA_ICTLR) Bits (Base Interrupt = 160)
31	SDMMC4	I2C6	-	-	DPAUX	
30	-	CLDVFS	-	CAR	GPU_NONSTALL	
29	ARB_SEM_GNT_CPU	-	SPI4	GPIO8	GPU_STALL	
28	ARB_SEM_GNT_COP	APB_DMA_COP	I2C3	WDT_AVP	TMR0	
27	-	SPI1	-	WDT_CPU	TMR9	
26	APB_DMA_CPU	SE	UARTD	-	TMR8	
25	FC_INT	USB3_DEV_PME	GPIO7	TMR5	TMR7	
24	PMC_INT	USB3_DEV_SMI	-	I2C4	TMR6	
23	SATA_CTL	GPIO5	GPIO6	APB_DMA_CH15	SDMMC4_SYS	
22	-	-	PMU_EXT	APB_DMA_CH14	SDMMC3_SYS	
21	USB2	I2C5	-	APB_DMA_CH13	SDMMC2_SYS	
20	USB	-	I2C2	APB_DMA_CH12	SDMMC1_SYS	
19	SDMMC3	EDP	SPI3	APB_DMA_CH11	CPU3_PMU_INTR	TMR13
18	-	TSEC	SPI2	APB_DMA_CH10	CPU2_PMU_INTR	TMR12
17	VII2C	XUSB_PADCTL	HDA	APB_DMA_CH9	CPU1_PMU_INTR	TMR11
16	VGPIO_INT	THERMAL	TSECB	APB_DMA_CH8	CPU0_PMU_INTR	TMR10
15	SDMMC2	-	-	APB_DMA_CH7	APB_DMA_CH31	MSELECT_ERROR
14	SDMMC1	UARTC	EMC	APB_DMA_CH6	APB_DMA_CH30	MPCORE_CTIIRQ3
13	SATA_RX_STAT	ACTMON	MC	APB_DMA_CH5	APB_DMA_CH29	MPCORE_CTIIRQ2
12	-	USB3_DEV_HOST	SOR	APB_DMA_CH4	APB_DMA_CH28	MPCORE_CTIIRQ1
11	DPAUX_INT1	USB3_HOST_PME	HDMI	APB_DMA_CH3	APB_DMA_CH27	MPCORE_CTIIRQ0
10	QUAD_SPI	TMR4	DISPLAYB	APB_DMA_CH2	APB_DMA_CH26	TMR_SHARED
9	NVDEC	TMR3	DISPLAY	APB_DMA_CH1	APB_DMA_CH25	FLOW_RSM_COP
8	NVJPEG	USB3_HOST_SMI	VIC	APB_DMA_CH0	APB_DMA_CH24	FLOW_RSM_CPU
7	SHR_SEM_OUTBOX_EMPTY	USB3_HOST_INT	ISP	APE_INT0	APB_DMA_CH23	-
6	SHR_SEM_OUTBOX_FULL	I2C	ISPB	APE_INT1	APB_DMA_CH22	-
5	SHR_SEM_INBOX_EMPTY	UARTB	VI	AVP_CACHE	APB_DMA_CH21	-
4	SHR_SEM_INBOX_FULL	UARTA	NVENC		APB_DMA_CH20	EVENT_GPIO_C
3	CEC	GPIO4	HOST1X_GEN_CPU	PCIE_MSI	APB_DMA_CH19	EVENT_GPIO_B
2	RTC	GPIO3	HOST1X_GEN_COP	PCIE_INT	APB_DMA_CH18	EVENT_GPIO_A
1	TMR2	GPIO2	HOST1X_SYNCPT_CPU	-	APB_DMA_CH17	MPCORE_INTERRIRQ
0	TMR1	GPIO1	HOST1X_SYNCPT_COP	DTV	APB_DMA_CH16	MPCORE_AXIERRIRQ

3.4.2 Arb_gnt Specific Interrupt Controller Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Arbitration semaphores provide a mechanism by which the two processors can arbitrate for the use of various resources. These semaphores provide a hardware locking mechanism, so that when a processor is already using a resource, the second processor is not granted that resource. There are 32 bits of Arbitration semaphores provided in the system.

The hardware does not enforce any resource association to these bits. It is left to the firmware to assign and use these bits.

The Arbitration Semaphores can also generate an interrupt when a hardware resource becomes available. The registers in this module configure these interrupts. When a 1 is set in the corresponding bit position of the Arbitration Semaphore Interrupt Source Register (CPU_enable or COP_enable), an interrupt will be generated when the processor achieves Grant Status for that resource.

The current Grant status can be viewed in the CPU_STATUS or COP_STATUS registers.

3.4.2.1 PRI_ICTLR_ARBGNT_CPU_STATUS_0

CPU Arbitration Semaphore Interrupt Status Register

Offset: 0x40 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GNT31_GNG0: Each bit is set by hardware when the corresponding arbitration semaphore ownership is granted to the CPU. Interrupt is cleared when the CPU writes the ARB_SMP.PUT register with the corresponding bit set.

3.4.2.2 PRI_ICTLR_ARBGNT_CPU_ENABLE_0

CPU Arbitration Semaphore Interrupt Enable Register

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	GER31_GER0: Writing a 1 in any bit position will enable the corresponding arbitration semaphore interrupt.

3.4.2.3 PRI_ICTLR_ARBGNT_COP_STATUS_0

BPMP-Lite Arbitration Semaphore Interrupt Status Register

Offset: 0x48 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GNT31_GNG0: Each bit is set by hardware when the corresponding arbitration semaphore ownership is granted to the BPMP-Lite. The interrupt is cleared when the BPMP-Lite writes the ARB_SMP.PUT register with the corresponding bit set.

3.4.2.4 PRI_ICTLR_ARBGNT_COP_ENABLE_0

BPMP-Lite Arbitration Semaphore Interrupt Enable Register

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	GER31_GER0: Writing a 1 in any bit position will enable the corresponding arbitration semaphore interrupt.

CHAPTER 4: SEMAPHORES

4.1 Arbitration Semaphores

Arbitration semaphores provide a mechanism by which two processors can arbitrate for the use of various resources.

These semaphores provide a hardware locking mechanism to ensure that when a processor is already using a resource, the second processor is not granted that resource. There are 32 bits of arbitration semaphores provided in the system. The hardware does not enforce any resource association to these bits and it is left to the user to assign and use these bits.

Any processor that needs to access a particular resource will request for the corresponding bit in the arbitration semaphores by writing a one to that bit in the Arbitration Semaphore Request register (SMP_GET register). Firmware will then check the corresponding bit in the Semaphore Granted Status register (SMP_GNT_ST register). If the requesting processor has been granted the resource, then the status returned will be a one.

Alternately, the processor can configure the interrupt controller to generate an interrupt when the resource becomes available. When the processor has finished using the resource, it releases the resource by writing a one to the corresponding bit in the Arbitration Semaphore Put Request register (SMP_PUT register). Additionally, pending request status is provided through the Arbitration Request Pending Status register (SMP_REQ_ST register).

(Note that an alternative mechanism to this exists in the Atomics block. Refer to the corresponding section of this document for further details. At the time of this writing, NVIDIA software does not make use of the Atomics block.)

4.2 Shared Semaphores

Shared semaphores are formed by three registers, each 32 bits wide:

- A read-only register showing the current value of the semaphore
- Two write-only registers to set and clear individual semaphore bits

These semaphores can be used to synchronize processors acting in a producer/consumer relationship; that is, a pair of them can be used to implement either a two-way or four-way handshake.

Correct operation of the shared semaphores requires to statically allocate an owner for each bit, because it is impossible to implement an atomic test and set (or similar operation) given the register interface. It also means that calling them semaphores is a little bit of stretch, they are simply atomic set and atomic clear registers.

4.3 Shared Mailboxes (Message Passing Elements)

This is limited to two instances of a single element mailbox, with an associated empty/not full status flag and the capability to generate interrupts.

The naming implies a specific direction of transfer (SHRD_INBOX and SHRD_OUTBOX), with the CPU as the reference to establish direction; that is, IN refers to the COP to CPU direction. This distinction is in fact enforced by hardware because the TAG bit semantic is dependent on the source of a transaction: TAG is write 1 clear for CPU (IN) or COP (OUT).

All message passing is via one register only, which contains 28 bits of uncommitted data (with a suggested usage), two bits controlling the generation of interrupt, and a field operating as status flag or a control bit depending on the direction of the transaction.

4.4 Semaphore Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

4.4.1 Arbitration Semaphore Registers

4.4.1.1 ARB_SEMA_SMP_GNT_ST_0

Semaphore Granted Status Register

Offset: 0x0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ARB_31_ARB_0: A one in any bit indicates that the processor reading this register as granted status for that bit. A zero indicates semaphore not granted.

4.4.1.2 ARB_SEMA_SMP_GET_0

Request Arbitration Semaphore Register

Offset: 0x4 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	GET_31_GET_0: Writing a one in any bit is a request for that semaphore bit by the processor performing the register write.

4.4.1.3 ARB_SEMA_SMP_PUT_0

Arbitration Semaphore Put Request Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PUT_31_PUT_0: Writing a one in any bit will clear the corresponding semaphore bit by the processor performing the register write.

4.4.1.4 ARB_SEMA_SMP_REQ_ST_0

Arbitration Request Pending Status (1=PENDING) Register

Offset: 0xc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REQ_31_REQ_0: A one in any bit indicates a request pending status. The corresponding bits are set when the request for the individual resource is pending. The read by the CPU of this register shows the pending status for the CPU and a read of this register by BPMP-Lite shows the pending status for BPMP-Lite.

4.4.2 Shared Resource Semaphore Registers

4.4.2.1 RES_SEMA_SHRD_SMP_STA_0

Shared Resource Semaphore Status

Offset: 0x0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SMP_31_SMP_0: SMP.27:SMP.24: Available in APB_DMA.REQUESTORS register

4.4.2.2 RES_SEMA_SHRD_SMP_SET_0

Shared Resource Semaphore Set-Bit Request

Offset: 0x4 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SET_31_SET_0: Semaphore set register. Writing a one to any bit will set the corresponding semaphore bit. Shared resource set-bit requests.

4.4.2.3 RES_SEMA_SHRD_SMP_CLR_0

Shared Resource Semaphore Clr-bit Request Register

Offset: 0x8 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CLR_31_CLR_0: Corresponding semaphore bit.

4.4.2.4 RES_SEMA_SHRD_INBOX_0

Shared Resource Inbox (Messages from COP (ARM7) to CPU)

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b000x0000000000000000000000000000)

Bit	Reset	Description
31	0x0	IE_IBF: Interrupt CPU on INBOX Full (TAG=1) 0 = EMPTY 1 = FULL
30	0x0	IE_IBE: Interrupt COP on INBOX Empty (TAG=0) 0 = EMPTY 1 = FULL
29	0x0	TAG: Set when the COP writes this register and cleared when CPU writes this register with this bit set to 1 (write one clear). 0 = INVALID 1 = VALID
27:24	0x0	IN_BOX_STAT: General-purpose data bits, suggested usage is for INBOX status (software can change the definition)
23:17	0x0	IN_BOX_CMD: General-purpose data bits, suggested usage is for INBOX command (software can change the definition)
16:0	0x0	IN_BOX_DATA: General-purpose Inbox data bits, suggested usage is for INBOX data (software can change the definition)

4.4.2.5 RES_SEMA_SHRD_OUTBOX_0

Shared Resource Outbox (Messages from CPU to COP (ARM7))

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b000x0000000000000000000000000000)

Bit	Reset	Description
31	0x0	IE_OBF: Interrupt COP on OUTBOX Full (TAG=1) 0 = EMPTY 1 = FULL
30	0x0	IE_OBE: Interrupt CPU on OUTBOX Empty (TAG=0) 0 = EMPTY 1 = FULL
29	0x0	TAG: Set when the CPU writes this register and cleared when the COP writes this register with this bit set to 1 (write one clear). 0 = INVALID 1 = VALID
27:24	0x0	OUT_BOX_STAT: General-purpose data bits, suggested usage is for Outbox OUTBOX status (software can change the definition)

Bit	Reset	Description
23:17	0x0	OUT_BOX_CMD: General-purpose data bits, suggested usage is for Outbox OUTBOX command (software can change the definition)
16:0	0x0	OUT_BOX_DATA: General-purpose Outbox data bits, suggested usage is for OUTBOX data (software can change the definition)

CHAPTER 5: CLOCK AND RESET CONTROLLER

The Clock and Reset (CAR) block contains all the logic needed to control most of the clocks and resets to the Tegra® X1 device. The CAR block provides the registers to program the PLLs and controls most of the clock source programming, and most of the clock dividers.

Power on reset (SYS_RESET_N_) is the primary reset for the Tegra X1 chip and is provided by the external PMIC.

5.1 Functional Description

Because this chip is the system controller, resets are generated in hardware automatically as part of the power-on (POR) or system (either hardware or software) reset sequence.

In POR, all blocks will be held in reset (with clocks disabled) except the minimal set of modules that are needed for system boot-up. At POR, the appropriate bits in RST_DEVICES_L/H/U/V/W/X/Y registers are set automatically by hardware. A “1” in the bit position signifies that block will be held at reset after POR. A “0” in the bit position signifies that block will have its reset de-asserted after POR.

Similarly for clocks, the appropriate bits in CLK_OUT_ENB_L/H/U/V/W/X/Y registers are set automatically by hardware. A “1” in the bit position signifies that block will have clock running during and after POR. A “0” in the bit position signifies that block will not have clock running during or after POR.

The blocks necessary for boot (known as boot blocks) include:

- The ARM7 Processor (BPMP-Lite) and its L1 cache, IRAM
- All system buses (PPSB, AHB, APB, etc.)
- Timer
- RTC
- eFUSE
- GPIO
- CoreSight

Each of the boot block devices will have their reset de-asserted at the end of the POR period (as well as their clocks enabled and use the Oscillator clock for their clock source). Boot blocks clock dividers are all set to divided-by-one.

During POR or system reset, the reset controller will de-assert reset to the boot blocks first and extend the resets to the CPU/ARM7 for another 256 oscillator clock periods. This will prevent either processor from talking to a boot device while it is still in the reset state.

Releasing a non-boot block/device from reset to bring into operation will require software to initiate a carefully controlled sequence with clock and reset control registers. This sequence must be implemented by software precisely to ensure correct operation of the hardware.

5.1.1 Notes

- Unless noted elsewhere, all modules support changing the clock divider ratio without disabling the clock.
- Unless noted elsewhere, all modules' clock switching is glitch-free except “audio_sync_clk”.
- For "audio_sync_clk", the clock source select needs to be set up before changing the device clock source to use that audio clock or to enable the device which use that audio sampling clock.
- At the time of writing, only 38.4 MHz is supported by NVIDIA for a crystal (or external clock) frequency. Other frequencies mentioned in this section are not supported.

- Before stopping the clock (via CLK_OUT_ENB_L/H/U/V/W/X/Y registers) and/or asserting reset (via RST_DEVICES_L/H/U/V/W/X/Y registers) to a module, first check the module to make sure it is not active. Stopping clock/asserting reset while the module is still busy can cause relatively minor problem such as incorrect data read/written, or catastrophic problem such as system hang. To ensure a module is not active, (a) disable the module by programming its disable bit if not already done so, and (b) wait until the module is not active by checking for its busy bit, done bit, count, or similar mechanism.

5.1.2 Clk_m Divisor

Clk_m which is generated from the oscillator pad input to the AP is used as the default option at which clocks startup in most of the clock multiplexors. It is also used to clock all the logic in CAR module (Example: The FSMs in switches that ensure that a change in switch parameters is glitch-less at the clock output).

For supporting various crystal frequencies without compromising on ULP use-cases, there is a special glitch-less divider in clk_m path which is capable of doing /1, /2, /3 and /4. A specific sequence is required for going to the divisor value that is needed. The default value of CLK_M_DIVISOR is to do a div1 (value = 0). The sequence to get to a divisor value that will work is:

Sequence for div1to div 3:

div1(reg value = 0) [RESET value] -> div4(reg value = 3) -> div3(reg value = 2)

Sequence for div1 to div2:

div1(reg value = 0) [RESET value] -> div4(reg value = 3) -> div2(reg value = 1).

Sequence for div1 to div4:

div1(reg value = 0) [RESET value] -> div4(reg value = 3)

The expectation is that this CLK_M_DIVISOR will only be changed once after powering VDD_SOC on in cold boot/LP0 exit path. So these sequences are verified with an oscillator clock of 38.4 MHz; div2 setting of the CLK_M divisor must be used. The result is 19.2 MHz clk_m.

Note: *Div2 is the recommended clk_m divisor value. Do not use any other value.*

Several register values in other IP blocks are dependent on the clk_m rate. Refer to the following sections for more information:

- The “TIMERUS_USEC_CFG_0” section of Chapter 8: Timers.
- The parts of Chapter 34: PCI Express (PCIe) Controller that mention “CLK25M” (CLK25M is driven by clk_m).

Common configuration for PLLs with fixed input divider policy:

- Always set fixed M-value based on the reference rate
- Always set P-value 1:1 for output rates above VCO minimum
- Choose minimum necessary P-value for output rates below VCO maximum
- Calculate N-value based on selected M and P
- Calculate SDM_DIN fractional part

5.1.3 Clock Generation Logic for SDMMC

There is a single clock switch for generating the timeout clock for all instances of SDMMC controllers(SDMMC1/SDMMC2/SDMMC3/SDMMC4). The timeout clock is set according to the timeout requirement of SDMMC controllers and is typically fixed to a frequency. This legacy timeout root clock is controlled like any other first level clock using CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0 which chooses the clock source and divisor. This root clock can be gated using control CLK_ENB_SDMMC_LEGACY_TM.

Apart from this root clock gate, there is a second level clock gate for this legacy timeout clock which goes to specific SDMMC controllers. The specific clock branch to SDMMCx controller is enabled by a logical expression which takes into account the following:

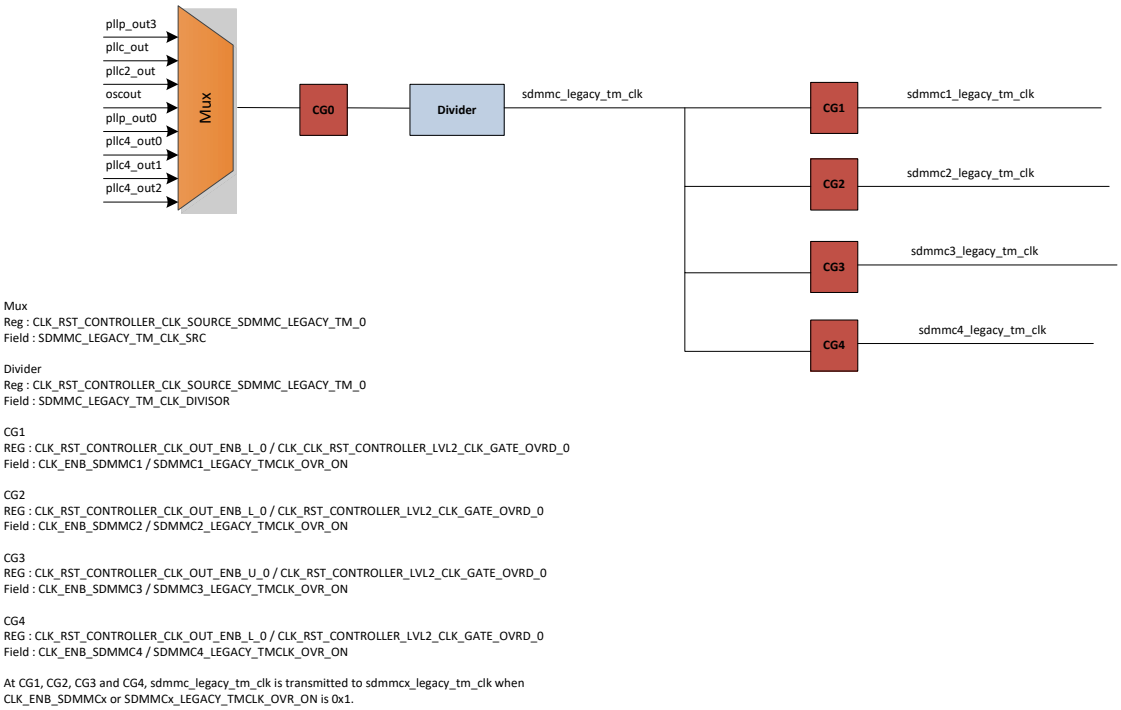
- The first level clock for the corresponding SDMMC controller [CLK_ENB_SDMMCx] AND the signal from the SDMMCx controller indicating the need for this clock are enabled.
- [OR] Override control in CAR to ignore all clock gating conditions [SDMMCx_LEGACY_TMCLK_OVR_ON].

Software does not need to program SDMMC1_LEGACY_TMCLK_OVR_ON==1 to get the clock running. It should be kept to '0' (default) only.

As long as CLK_ENB_SDMMC1==1, sdmmc1_legacy_tmclk will carry to SDMMC1 controller.

Similarly, if CLK_ENB_SDMMC2==1, sdmmc2_legacy_tmclk will carry to SDMMC2 controller.

Figure 6: SDMMC Timeout Clock Frequency



5.1.4 Audio Sync Clock

The audio sync clock synchronizes communication between 2 audio devices to prevent a FIFO overrun/underrun condition in either of the audio devices. For example, one can receive data from SPDIFIN and transmit the same data back out to an I2S speaker. But if the SPDIFIN stream is 43.9 kHz while transmitting a 44.1 kHz sample rate to I2S, the SPDIFIN receive FIFO can get overrun while the I2S transmit FIFO can get underrun.

> ***Note:*** *There is no clock switching protection for the audio sync clock so one needs to set up the desired clock source before enabling the audio transmit/receive device.*

5.1.5 Analog PLLs

The following PLLs are analog:

- PLLP
- refPLLE

- PLLU
- PLLA
- PLLE
- PLLC4
- UTMIP
- PLLM
- PLLD
- PLLD2, PLLDP
- PLLX
- PLLG

To start the PLL:

1. Change IDDQ from 1 to 0.
2. Wait 5 µs (10 µs in case of UTMIPLL). If wait is less, then PLL lock time can be higher than the specification.
3. Program PLL registers (when IDDQ=0 and ENABLE=0).
4. Enable 0 →1 (clock-in must be running before asserting ENABLE).
5. Wait for LOCK assertion to use PLL [PLL_STATE=LOCKED].

To stop the PLL:

6. Use ENABLE 1→0 to stop [PLL_STATE=DISABLED, Lower latency wake up]. Go to Step 4 to turn it back on.
7. With ENABLE=0, change IDDQ from 0→1 for better power savings [PLL_STATE=IDDQ]. Go to Step 1 to turn it back on.
8. Use reference clock enable controls to save more power but make sure to turn it back on before powering it up.

Notes:

- *The core voltage rail can be brought down any time after Step 6 but ideally after Step 7. The PLL inputs will be clamped and the PLL will be in deep power down mode. All the PLL controls will be reset to their defaults when the rail is powered back up and so re-initialization will start from Step 1.*
- *PLL analog voltage rails are expected to be turned off in LP0 state in Tegra X1. This is different to prior Tegra chips.*
- *PLLM can be auto-started when powering up from LP0. The sequencing is taken care in hardware.*

5.1.5.1 Dynamic ramp and fractional NDIV [Where feature available in PLL]

1. Program NDIV_NEW, SDM_DIN_NEW to target frequency. Program DYNRAMP_STEPA and DYNRAMP_STEPB.
2. Set EN_DYNRAMP=1, PLL starts to acquire lock to target frequency.
3. PLL asserts DYNRAMP_DONE
4. Program SDM_DIN= SDM_DIN_NEW, NDIV=NDIV_NEW
5. Set EN_DYNRAMP=0

5.1.5.2 Spread using SDM

- PLLD2

- PLLDP

Program SDM-SSC controls and set EN_SSC to 1 before using PLL.

5.1.5.3 Spread using SDA

- PLLE

For PLLE SDA

To turn on Spread using SSA:

1. Load Spread Coefficients
2. BYPASS_SS=0
3. SSC_BYP=0
4. Wait 300nS and INTERP_RESET=0 At this stage spread spectrum is enabled

To turn OFF spread spectrum:

1. BYPASS_SS=1
2. INTERP_RESET=1, SSC_BYP=1

5.1.6 Hybrid PLLs (HPLLs)

The following PLLs are hybrid:

- PLLC
- PLLA1
- PLLC2
- PLLC3

To start the PLL:

1. Assert RESET if not already asserted.
2. Program PLL registers.
3. IDDQ 1 ->0. Wait for a 25 cycles of reference clock.
4. Change RESET from 1->0.
5. Change ENABLE from 0->1.
6. Wait for frequency and phase LOCK to start using the PLL.

To stop the PLL:

1. Change ENABLE control to 0 and RESET to 1. Back to step 4 to turn ON PLL.
2. Optionally engage the reference clock gating control to save some power. It is to be enabled prior to turning the PLL back on.
3. Change IDDQ to 1 for better power savings.
4. Change frequency dynamically.
5. Use LOOP_CTRL=0 or LOOP_CTRL=1 for major/minor changes in frequency. Change NDIV/fractional NDIV parameter for the change in frequency.
6. Wait for frequency and phase LOCK for completion of sequence in hardware.

5.1.7 Generic Clock Switch Programming Sequence

To set up a non-boot device for operation (only apply if a device has a CLK_SOURCE_<mod> register)

1. Make sure the device's reset is asserted (via RST_DEVICES_L/H/U/V/W/X/Y registers).
2. Enable clock to the device (via CLK_OUT_ENB_L/H/U/V/W/X/Y registers).
3. Change the clock divisor to the device (via CLK_SOURCE_ register).
4. Wait 2 µs to make sure the clock divider has changed.
5. Change the clock source to the device (via the CLK_SOURCE_<mod> register).
6. Wait 2 µs to make sure clock source/device logic is stabilized.
7. De-assert the device's reset (via RST_DEVICES_L/H/U/V/W/X/Y registers).

To change a device's clock divider and/or source after boot-up (only apply if a device has a CLK_-SOURCE_<mod> register):

(A) Method 1 -- (using reset).

1. Make sure the device is disabled (via the device's register).
2. Assert device's reset (via RST_DEVICES_L/H/U/V/W/X/Y registers).
3. Make sure clock to the device is enabled (via CLK_OUT_ENB_L/H/U/V/W/X/Y registers).
4. Change the clock divisor to the device (via CLK_SOURCE_<mod> register).
5. Wait 2 µs to make sure the clock divider has changed.
6. Change the clock source to the device (via the CLK_SOURCE_<mod> register).
7. Wait 2 µs to make sure clock source/device logic is stabilized.
8. De-assert the device's reset (via RST_DEVICES_L/H/U/V/W/X/Y registers).

(B) Method 2 -- (not using reset when unit is running).

1. Make sure the clock to the device is enabled (via CLK_OUT_ENB_L/H/U/V/W/X/Y registers).
2. Depending on the maximum rated frequency of the device and the current and target clock source/divider value, either change the divider first or the clock source first to avoid a temporary situation where the maximum frequency for that device is violated. Make sure to wait for 2 µs between changing the divider and clock source programming.).
3. Wait 2 µs to make sure the device logic is stabilized.

De-assert reset to device (without need to change clock) after boot-up:

1. Make sure the device's reset is asserted (via RST_DEVICES_L/H/U/V/W/X/Y registers).
2. Wait 2 µs to make sure the device logic is stabilized.
3. De-assert the device's reset (via RST_DEVICES_L/H/U/V/W/X/Y registers).

Method used to wait for 2 µs:

1. To use one of the four timers, refer to Chapter 8: Timers of this document.
2. Program the TMR_PTV register's TMR_PTV field with the desired microsecond count.
3. Program the TMR_PTV register's EN field to enable the timer.
4. Poll the TMR_PCR register's TMR_PCV field until it reaches 0 to indicate the microsecond count has been reached.
5. Program the TMR_PTV register's EN field to disable the timer.

5.1.7.1 Clock Configuration for Each Peripheral

In general, each block or module has a dedicated CLK_SOURCE_ register that provides clock source and clock divider control to that device. The clock source select is typically 2 bits long to select one of the four clock sources. The divider is typically 8 bits

long, consisting of a 7-bit integer and a 1-bit fraction. Furthermore, depending on the specific module, the CLOCK_SOURCE_ register may contain additional control bits.

Note: *The 16-bit UARTA/UARTB/UARTC clock divider control is provided by the UART register set and not by the CLK_SOURCE_ registers. To switch from one clock source to the next, both clock sources must be active/running.*

5.1.8 Main Clock Sources Used by the System:

(A) Primary clocks.

- "osc" - 38.4 MHz in Tegra X1 platforms (At the time of writing, only 38.4 MHz is supported by NVIDIA for a crystal (or external clock) frequency. Other frequencies mentioned in this section are not supported.)
  - "clk_m" - Derived from above and run at 19.2 MHz
  - "clk_s" – 32.768 kHz clock from system PMIC

(B) PLL clocks.

Notes:

- *Before reprogramming a clock source to generate a higher rate, software must first ensure that the voltage regulator that supplies the clock source and any downstream devices is set to at least the minimum required voltage for that rate. Similarly, after a clock source is reprogrammed to generate a lower rate, it may be possible for software to program the voltage regulator to reduce the voltage supplied to the rail to save energy. Information on the appropriate minimum voltage levels can be obtained from NVIDIA.*
- *For PLLCx, programming/sequencing is performed by the clocking software. All settings applied by the central clock driver are provided by NVIDIA. If the customer's software is developed from scratch by the customer (not based on NVIDIA kernel with NVIDIA clock driver), they can potentially redefine PLLCx configurations and usage. However, user-defined configurations are not guaranteed to be part of the chip characterization. Only configurations set by the NVIDIA clock software are guaranteed to be characterized by NVIDIA.*
- *All pad PLLs are managed by IP drivers.*

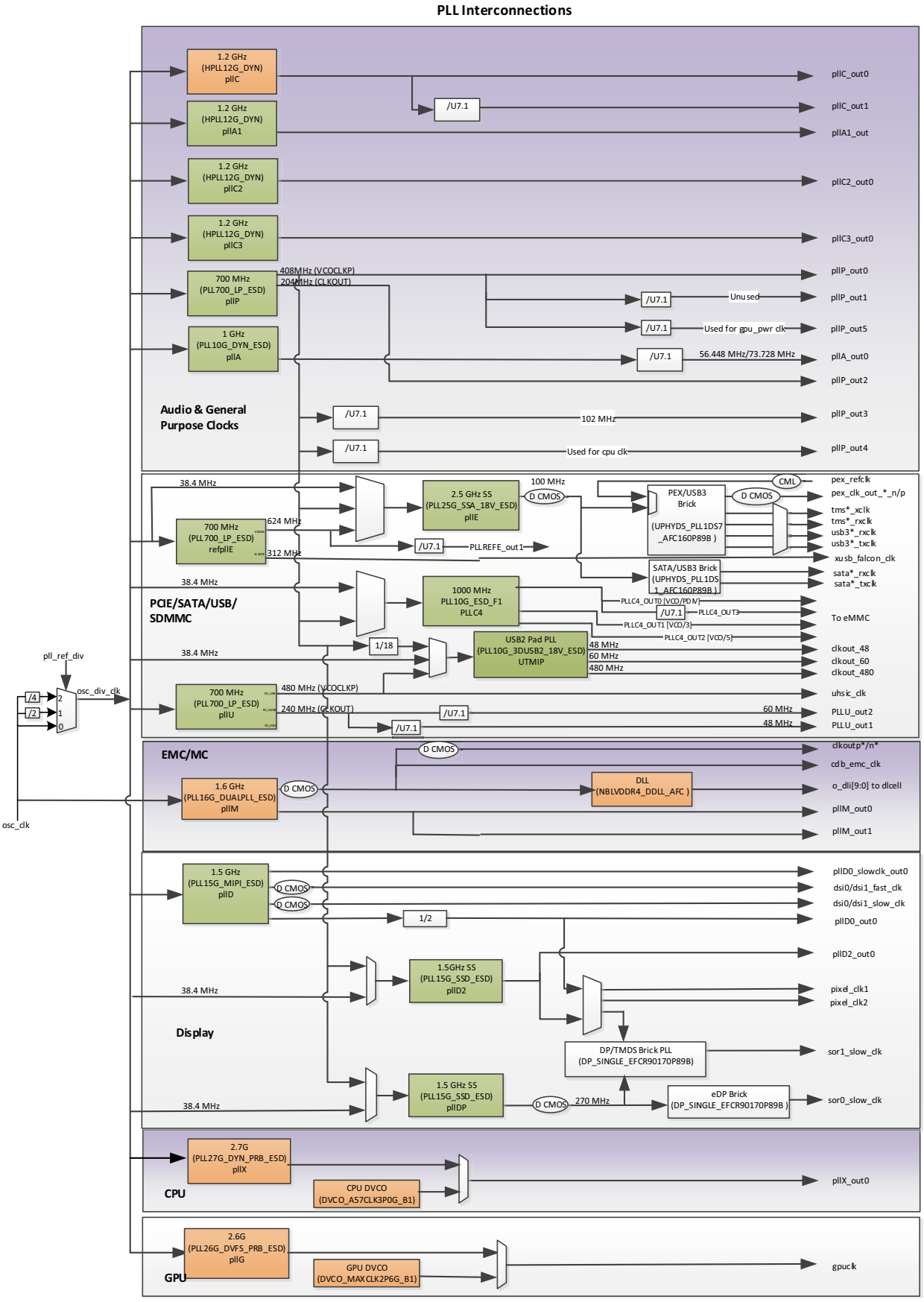
- "PLLC" is general purpose and its output is called "pllC_out". PLLC is for camera subsystem. The frequency is programmed by the clocking software based on the frequency requests from camera drivers delivered by NVIDIA. This frequency is a function of the sensor and the camera use case.
- "PLLC2" is general purpose and its output is called "pllC2_out". PLLC2 is shared between SE, VIC, TSECB, NVJPG and is set according to the requirement from these drivers by Clocking software.
- "PLLC3" is general purpose and its output is called "pllC3_out". PLLC3 is shared between NvENc, NvDec and is set according to the requirement from these drivers by Clocking software.
- "PLLC4" has 3 outputs: PLLC4_out0/1/2: OUT0=VCO/PDIV, OUT1=VCO/DIV3, OUT2=VCO/DIV5. Primarily used for SD/eMMC and also a source for system buses.
- "PLLM" is a composite clock source that supplies a clock for the DRAM interface and controller. PLLM consists of two symmetric, internal PLLs, PLLMA and PLLMB, and a multiplexer that selects which internal PLL provides the PLLM clock output. The dual-PLL architecture provides a stable, high-frequency clock source to system memory during DVFS operations.
- "PLLP" (peripheral) has a fixed frequency and its output is called "pllP_out0". This runs at 408 MHz and is the VCO output of PLLP.
- "PLLA" (audio) is used for audio purposes, generating precise codec sampling frequencies.
- "PLLA1"(audio) is used for clocking the ADSP and its output is called "PLLA1_out".

- “PLLU” (USB) has two fixed outputs at 480 MHz for HSIC and 240 MHz used for other USB clocks.
- “PLLD” (DSI) is primarily used for the display and its output PLLD_out. After a /2 fix is called “pllD_out0”.
- “PLLD2”(display) is primarily used for the display and its output (after a /2 fix) is called “pllD2_out0”.
- “PLLDP” (display port) is primarily used for supplying the reference clock to display port SOR brick. Its output is called “pllDP_out0”.
- “PLLX” has extra high frequency used by CPUs and is called “pllX_out0”. It is the primary source for Slow CPU but not for Fast CPU as it is not voltage noise-aware. (Note: A53 cluster has PLLX as the primary source.) There is no other PLL but we have DVCO-based voltage noise aware clocking source called CLDVFS.
- “refPLLE” is only used by PLLE, XUSB Falcon, CoreSight debug clocks.
- “PLLE” provides spread spectrum for PCIe, SATA, USB3 (if they are present). PLLE output frequency must be 100 MHz.
- UTMIPLL is for USB2.0 and its output frequency must be fixed at 480 MHz. Note: UTMIPLL is initially configured by clock driver, but in case of USB2 UTMIPLL run-time IDDQ mode is under USB2 driver control.

(C) PLL divided down clocks (each divider has 7 integer bits and 1 fractional bit).

- “pllC_out1” is divided-down from “pllC_out0”.
- “pllM_out1” is divided-down from “pllM_out0”.
- “pllP_out1” is divided-down from “pllP_out0”.
- “pllP_out2” is divided-down from “pllP_VCO”. This is done using the divP of the PLLP. DIVP is not glitch-free. So have to ensure that out2 is not used before it can be changed.
- “pllP_out3” is divided-down from “pllP_out0”.
- “pllP_out4” is divided-down from “pllP_out0”.
- “pllP_out5” is divided-down from “pllP_out0”.
- “pllA_out0” is divided-down from the “pllA” output.

Figure 7: Clocking Sources



PLLP Configuration Information

PLLP is configured as a fixed frequency PLL. The code in the Boot ROM configures PLLP at a fixed 408 MHz with help from hardware by programming osc_freq. Later on, software (typically the Boot Loader) may change this to another “fixed” frequency.

Table 16: Boot ROM Configuration to 408 MHz

Reference Frequency	38.4 MHz	19.2 MHz	12.0 MHz	48.0MHz	Resulting Frequency
DIVN	085 (055h)	085 (055h)	034 (022h)	034 (022h)	-
DIVM	08 (08h)	04 (04h)	01 (01h)	04 (04h)	-
DIVP	1 (1h)	1 (1h)	1 (1h)	1 (1h)	PLLP_out2 = 204 MHz
KVCO	0 (0h)	4 (4h)	8 (8h)	8 (8h)	-
KSETUP	0 (0h)	0 (0h)	0 (0h)	0 (0h)	-
KCP	0 (0b)	0 (0b)	0 (0b)	0 (0b)	-
OUT1 div ratio	42.5 (0dh)	42.5 (0dh)	42.5 (0dh)	42.5 (0dh)	9.6 MHz
OUT3 div ratio	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	102.0 MHz
OUT4 div ratio	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	102.0 MHz
OUT5 div ratio	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	204.0 MHz

Table 17: PLLU Configuration Information (Reference Clock osc_div_clk and Outputs fixed at 12 MHz/48 MHz/60 MHz/480 MHz

Reference Frequency	38.4 MHz	19.2 MHz	12.0 MHz	48.0MHz	Resulting Frequency
DIVN	025 (019h)	025 (019h)	040 (028h)	040 (028h)	
DIVM	02 (02h)	01 (01h)	01 (01h)	04 (04h)	
DIVP	02 (01h)	02 (01h)	02 (01h)	02 (01h)	240 MHz at VCO/ DIVP
KCP	0 (0h)	0 (0h)	0 (0h)	0 (0h)	
KVCO	0 (0h)	0 (0h)	0 (0h)	0 (0h)	
SETUP	0 (0h)	0 (0h)	0 (0h)	0 (0h)	
OUT1 div ratio	5 (08h)	5 (08h)	5 (08h)	5 (08h)	48 MHz
OUT2 div ratio	4.0 (06h)	4.0 (06h)	4.0 (06h)	4.0 (06h)	60 MHz

Notes:

- *Most of the PLLs unless specified have "osc_div_clk" as the reference clock. This is the same frequency as "osc":38.4 MHz: Does not need to be divided in Tegra X1. The option to divide down the osc clock is still present but not used for improving clock quality.*
- *At the time of writing, only 38.4 MHz is supported by NVIDIA for a crystal (or external clock) frequency. Other frequencies mentioned in this section are not supported.*

PLLM Programming Sequence During Core DVFS:

1. One of the PLLM* PLLs that is inactive is reprogrammed to generate the desired rate and enabled. (If the memory clock source is generated by PLLP, then either PLLMA or PLLMB can be used.)
2. The EMC clock glitchless mux is reprogrammed. This initiates a hardware sequence that switches the active PLLM clock source to the PLL that was programmed in step 1.
3. The PLLM* PLL that is now inactive is turned off to save energy.

PLL Programming Constraints:

In general, there are 3 requirements for each PLL with which software needs to comply:

- Input frequency range (REF).

- Comparison frequency range (CF). CF = REF/DIVM, where DIVM is the input divider control.
- VCO frequency range (VCO). VCO = CF * DIVN, where DIVN is the feedback divider control.

Note: *The final PLL output frequency (FO) = VCO/DIVP, where DIVP is the post divide control, which is encoded.*

All supported crystals are used by the PLLs directly, without any pre-divider.

Contact your NVIDIA representative for more details on the PLLs.

Table 18: PLDIV - Divider Control for CLKOUT

PLDIV[3:0]	CLKOUT
0000	vcoclock/1 (pdivider is powered down)
0001	vcoclock/2
0010	vcoclock/3
0011	vcoclock/4
0100	vcoclock/5
0101	vcoclock/6
0110	vcoclock/8
0111	vcoclock/10
1000	vcoclock/12
1001	vcoclock/16
1010	vcoclock/12
1011	vcoclock/16
1100	vcoclock/20
1101	vcoclock/24
1110	vcoclock/32
1111	Reserved

Note: *In this document, CPULP, SCPU, and Slow CPU are synonymous as are CPUG, FCPU, and Fast CPU. In this switched cluster implementation for Tegra X1, the switch used for generating the clocks is the same. Except for BURST_POLICY and SUPER_CCLK_DIVIDER, use the 'G' registers for both SCPU and FCPU control. For these two registers use the 'G' and 'LP' registers for FCPU and SCPU, respectively.*

5.1.9 CPU (CCLK) and BPMP-Lite/System (SCLK) Clock Control

In this section, MPCore A57/A53 is referred to as CPU while ARM7 is referred to as BPMP-Lite. In this context, CPU refers to the active CPULP or CPUG cluster. Each CCLK and SCLK clock domain can have 5 states: SUSP (suspend) where the clock source is 32 kHz, and normal states (IDLE, RUN, IRQ, FIQ).

Each of the normal states can be selected by software from 8 different clock sources. Furthermore, if any of the CPU/BPMP-Lite FIQ/IRQ bit is enabled, a hardware auto trigger feature will be enabled such that hardware will jump from any state to IRQ or to FIQ automatically. Of course, if the source is from a PLL, software needs to guarantee that the PLL clock is running and stable before changing clock sources.

There are many usage models that can be derived from this mechanism: For example:

- Software can keep changing the clock source to one state (i.e., CWAKEUP_IDLE_SOURCE) without changing the CPU_STATE field.
- Software can have the concept of multiple states by first setting up CWAKEUP__SOURCE and then changing the CPU_STATE field.

- Software can enable hardware auto detect of IRQ/FIQ to jump to the IRQ or FIQ state.

Note: *Whenever the clock source is switched, the clock is stopped for approximately 400-600 ns.*

5.1.10 CCLK and SCLK Super Clock Divider Control

The super clock divider allows a very fine tune of clock frequency going to CPU and BPMP-Lite/system using clock skipping technique. It is different from a traditional divider (1/n) in that both the numerator and denominator are programmable (m/n). Both the numerator and denominator are 8 bits each. Thus, “effective” frequency = source frequency * (m/n). Furthermore, if any of the CPU/BPMP-Lite FIQ/IRQ bits is enabled, hardware will auto disable the super clock divider functionality.

There are many usage models that can be derived from this mechanism. For example:

- There is no clock source switching penalty. Software can pick a PLL output as a maximum frequency source via CCLK/SCLK_BURST_POLICY and keep changing SUPER_CCLK/SCLK_DIVIDER to yield the desired lower “effective” frequency.
- For applications where the “osc” frequency is more than sufficient to do the job, PLLs can be turned off to save power and the super clock divider can further divide down the "osc" clock to yield even more power saving.
- If auto IRQ/FIQ feature is enabled, CCLK/SCLK will automatically jump back to full frequency to handle high priority interrupt routines.

Note: *From a dynamic voltage scaling (DVS) standpoint, the full clock frequency source (not the output frequency of the super clock divider) going into the super clock divider should be used to determine how low one can lower the voltage. If m > n, the resulting super clock divider output frequency will be the same as the input frequency. In other words, there will be no clock skip or divide down.*

5.2 Clock and Reset Controller Registers

Refer to Section 1.4 Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

5.2.1 CLK_RST_CONTROLLER_RST_SOURCE_0

WatchDog (Deadman) Timer

The watchdog timer is used to recover from hang/lockup condition by resetting the system and/or BPMP-Lite (ARM7) and/or CPU (MPCore). Either timer1 or timer2 can be used as watchdog timer. Refer to Chapter 8: Timers of this document for more information on watchdog timer usage.

Offset: 0x0 | Read/Write: R/W | Reset: 0x000XXX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00x000)

Bit	R/W	Reset	Description
19	RO	X	WDT_CPU3_RST_STA: CPU3 reset by watchdog timer (RO)
18	RO	X	WDT_CPU2_RST_STA: CPU2 reset by watchdog timer (RO)
17	RO	X	WDT_CPU1_RST_STA: CPU1 reset by watchdog timer (RO)
16	RO	X	WDT_CPU0_RST_STA: CPU0 reset by watchdog timer (RO)
13	RO	X	SWR_SYS_RST_STA: System reset by software (RO)
12	RO	X	WDT_SYS_RST_STA: System reset by watchdog timer (RO)
11	RO	X	SWR_COP_RST_STA: BPMP-Lite reset by software (RO)
10	RO	X	WDT_COP_RST_STA: BPMP-Lite reset by watchdog timer (RO)
9	RO	X	SWR_CPU_RST_STA: CPU reset by software (RO). This bit has been deprecated. Always returns 1'b0.
8	RO	X	WDT_CPU_RST_STA: CPU reset by watchdog timer (RO)
5	RW	DISABLE	WDT_EN: Enable WatchDog Timer (Dead Man Timer) 0 = DISABLE 1 = ENABLE
4	RW	TIMER1	WDT_SEL: WatchDog Timer Select 0 = TIMER1 1 = TIMER2
2	RW	DISABLE	WDT_SYS_RST_EN: Enable WatchDog Timer reset for system. 0 = DISABLE 1 = ENABLE
1	RW	DISABLE	WDT_COP_RST_EN: Enable WatchDog Timer reset for BPMP-Lite 0 = DISABLE 1 = ENABLE
0	RW	DISABLE	WDT_CPU_RST_EN: Enable WatchDog Timer reset for CPU 0 = DISABLE 1 = ENABLE

5.2.2 CLK_RST_CONTROLLER_RST_DEVICES_L_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x1cd3dXXX (0b0xx111xx11x1xx1111x1xx1x11xx100x)

Bit	R/W	Reset	Description
31	RW	DISABLE	SWR_CACHE2_RST: Reset BPMP-Lite cache controller. 0 = DISABLE 1 = ENABLE
28	RW	ENABLE	SWR_HOST1X_RST: Reset HOST1X. 0 = DISABLE 1 = ENABLE
27	RW	ENABLE	SWR_DISP1_RST: Reset DISP1 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
26	RW	ENABLE	SWR_DISP2_RST: Reset DISP2 controller. 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_ISP_RST: Reset ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	ENABLE	SWR_USBD_RST: Reset USB controller. 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	SWR_VI_RST: Reset VI controller. 0 = DISABLE 1 = ENABLE
17	RW	ENABLE	SWR_PWM_RST: Reset Pulse Width Modulator 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_SDMMC4_RST: Reset SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_SDMMC1_RST: Reset SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_I2C1_RST: Reset I2C1 Controller 0 = DISABLE 1 = ENABLE
9	RW	ENABLE	SWR_SDMMC2_RST: Reset SDMMC2 Controller 0 = DISABLE 1 = ENABLE
8	RO	X	SWR_GPIO_RST: Reset GPIO Controller 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SWR_UARTB_RST: Reset UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SWR_UARTA_RST: Reset UARTA Controller 0 = DISABLE 1 = ENABLE
5	RO	X	SWR_TMR_RST: Reset Timer Controller 0 = DISABLE 1 = ENABLE
3	RW	ENABLE	SWR_ISPB_RST: Reset ISPB controller
2	RW	DISABLE	SWR_TRIG_SYS_RST: Write 1 to pulse System Reset Signal. Hardware clears this bit. 0 = DISABLE 1 = ENABLE
1	RW	DISABLE	SWR_COP_RST: Write 1 to force BPMP-Lite Reset Signal. Software needs to clear this bit when done. 0 = DISABLE 1 = ENABLE
0	RO	X	SWR_CPU_RST: Tied to 0.

5.2.3 CLK_RST_CONTROLLER_RST_DEVICES_H_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x87d1f32X (0b1xxxx11111x1xxx11111xx110x1xx11x)

Bit	R/W	Reset	Description
31	RW	ENABLE	SWR_BSEV_RST: Reset BSEV controller. 0 = DISABLE 1 = ENABLE
26	RW	ENABLE	SWR_USB2_RST: Reset USB2 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
25	RW	ENABLE	SWR_EMC_RST: Reset EMC controller. 0 = DISABLE 1 = ENABLE
24	RW	ENABLE	SWR_MIPI_CAL_RST: Reset MIPI CAL Logic. 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_UARTC_RST: Reset UARTC controller 0 = DISABLE 1 = ENABLE
22	RW	ENABLE	SWR_I2C2_RST: Reset I2C2 controller. 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	SWR_CSI_RST: Reset CSI controller. 0 = DISABLE 1 = ENABLE
16	RW	ENABLE	SWR_DSI_RST: Reset DSI controller 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_I2C5_RST: Reset I2C5 controller 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_SPI3_RST: Reset SPI3 controller. 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_SPI2_RST: Reset SPI2 controller. 0 = DISABLE 1 = ENABLE
9	RW	ENABLE	SWR_SPI1_RST: Reset SPI1 controller. 0 = DISABLE 1 = ENABLE
8	RW	ENABLE	SWR_KFUSE_RST: Reset KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	ENABLE	SWR_STAT_MON_RST: Reset statistic monitor. 0 = DISABLE 1 = ENABLE
2	RW	ENABLE	SWR_APBDMA_RST: Reset APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	ENABLE	SWR_AHBDMA_RST: Reset AHB-DMA. 0 = DISABLE 1 = ENABLE
0	RO	X	SWR_MEM_RST: Reset MC. This bit is disabled for security reasons. You can write to it but it will always read as 0.

5.2.4 CLK_RST_CONTROLLER_RST_DEVICES_U_0

Offset: 0xc │ Read/Write: R/W │ Reset: 0x828ec5fa (0b1xxxxx1x1xxx111x11x0x10111111x1x)

Bit	Reset	Description
31	ENABLE	SWR_XUSB_DEV_RST: Reset XUSB DEV logic. 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_XUSB_HOST_RST: Reset XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_EMUCIF_RST: Reset EMUCIF logic. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	ENABLE	SWR_TSEC_RST: Reset TSEC logic. 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_DSIB_RST: Reset DSIB 0 = DISABLE 1 = ENABLE
17	ENABLE	SWR_I2C_SLOW_RST: Reset I2C_SLOW 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_DTV_RST: Reset DTV 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_SOC_THERM_RST: Reset SOC_THERM. 0 = DISABLE 1 = ENABLE
10	ENABLE	SWR_PCIEXCLK_RST: Reset PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	DISABLE	SWR_CSITE_RST: Reset CoreSight controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_AFI_RST: Reset AFI controller. 0 = DISABLE 1 = ENABLE
7	ENABLE	Unused, reserved.
6	ENABLE	SWR_PCIE_RST: Reset PCIe® controller. 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_SDMMC3_RST: Reset SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	ENABLE	SWR_SPI4_RST: Reset SPI4 controller. 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_I2C3_RST: Reset I2C3 controller. 0 = DISABLE 1 = ENABLE
1	ENABLE	SWR_UARTD_RST: Reset UARTD controller. 0 = DISABLE 1 = ENABLE

5.2.5 CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x80000130 (0b10x000xx00x0x00000x0000100110xx0)

Bit	Reset	Description
31	ENABLE	CLK_ENB_CACHE2: Enable clock to BPMP-Lite cache controller. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_I2S`: Enable clock to I2S` Controller 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_HOST1X: Enable clock to Host1x. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_DISP1: Enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_DISP2: Enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_ISP: Enable clock to ISP controller. 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_USBD: Enable clock to USB controller 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_VI: Enable clock to VI controller. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_I2S3: Enable clock to I2S3 controller 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_PWM: Enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_SDMMC4: Enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SDMMC1: Enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_I2C1: Enable clock to I2C1 Controller 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_I2S2: Enable clock to I2S2 Controller 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_SPDIF: Enable clock to S/PDIF Controller 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SDMMC2: Enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	CLK_ENB_GPIO: Enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_UARTB: Enable clock to UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_UARTA: Enable clock to UARTA Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	ENABLE	CLK_ENB_TMR: Enable clock to Timer Controller (sticky) 0 = DISABLE 1 = ENABLE
4	ENABLE	CLK_ENB_RTC: Enable clock to RTC Controller 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_ISPB: Enable clock – ISPB 0 = DISABLE 1 = ENABLE

5.2.6 CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000080 (0b0xxxx00000x0xxx00000xx00100xx000)

Bit	Reset	Description
31	DISABLE	CLK_ENB_BSEV: Enable clock to BSEV Controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_USB2: Enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_EMC: Enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
24	DISABLE	CLK_ENB_MIPI_CAL: Enable clock to MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_UARTC: Enable clock to UARTC controller 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_I2C2: Enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_CSI: Enable clock to CSI controller 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_DSI: Enable clock to DSI controller 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_ I2C5: Enable clock to I2C5 controller 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SPI3: Enable clock to SPI3 controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_SPI2: Enable clock to SPI2 Controller. 0 = DISABLE 1 = ENABLE
11	ENABLE	CLK_ENB_JTAG2TBC: Enable clock to jtag2tbc Interface. 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SPI1: Enable clock to SPI1 Controller. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_KFUSE: Enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	ENABLE	CLK_ENB_FUSE: Enable clock to FUSE controller and host2jtag clock. 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_PMC: Enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_STAT_MON: Enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
2	DISABLE	CLK_ENB_APBDMA: Enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_AHBDMA: Enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_MEM: Enable clock to MC. 0 = DISABLE 1 = ENABLE

5.2.7 CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x01f00200 (0b0000xx011111000x00x0xx1000000x0x)

Bit	Reset	Description
31	DISABLE	CLK_ENB_XUSB_DEV: Enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_DEV1_OUT: Enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_DEV2_OUT: Enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_SUS_OUT: Enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
26	ENABLE	CLK_M_DOUBLER_ENB: Enable CLK_M clock doubler
25	DISABLE	CLK_ENB_XUSB_HOST: Enable clock to XUSB HOST 0 = DISABLE 1 = ENABLE
24	ENABLE	CLK_ENB_CRAM2: Enable BPMP-Lite cache RAM clock. 0 = DISABLE 1 = ENABLE
23	ENABLE	CLK_ENB_IRAMD: Enable IRAMD clock. 0 = DISABLE 1 = ENABLE
22	ENABLE	CLK_ENB_IRAMC: Enable IRAMC clock. 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_IRAMB: Enable IRAMB clock. 0 = DISABLE 1 = ENABLE
20	ENABLE	CLK_ENB_IRAMA: Enable IRAMA clock. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_TSEC: Enable TSEC clock. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_DSIB: Enable clock to DSIB 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_I2C_SLOW: Enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_DTV: Enable clock to DTV controller. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SOC_THERM: Enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_LA: Enable clock to LA. 0 = DISABLE 1 = ENABLE
9	ENABLE	CLK_ENB_CSITE: Enable clock to CoreSight. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_AFI: Enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	DISABLE	Unused, reserved.

Bit	Reset	Description
6	DISABLE	CLK_ENB_PCIE: Enable clock to PCIe. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_SDMMC3: Enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_SPI4: Enable clock to SPI4. 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_I2C3: Enable clock to I2C3. 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_UARTD: Enable clock to UARTD. 0 = DISABLE 1 = ENABLE

5.2.8 CLK_RST_CONTROLLER_SUPER_CCLK_DIVIDER_0

Offset: 0x24 I Read/Write: R/W I Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = disable divider. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override. 0 = DISABLE 1 = ENABLE
30	DISABLE	SUPER_CDIV_USE_THERM_CONTROLS: 1 = use therm controls for pulse skipper (cpug only) 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: See 'Invert Duty-Cycle Distortion control'. 1 = Enable Inversion of DCD (duty-cycle distortion) 0 = Disable Inversion of DCD (duty-cycle distortion) 0 = DISABLE 1 = ENABLE
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = COP FIQ doesnt impact super clock divider enable. 1 = Disable super clock divider on COP FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = CPU FIQ doesnt impact super clock divider enable. 1 = Disable super clock divider on CPU FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = COP IRQ doesnt impact super clock divider enable. 1 = Disable super clock divider on COP IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = CPU IRQ doesnt impact super clock divider enable. 1 = Disable super clock divider on CPU IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: Divide by [(N/2)+1]
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override.

5.2.9 CLK_RST_CONTROLLER_SCLK_BURST_POLICY_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x10000000 (0b00010000xxxxxxxxx000x000x000x000)

Bit	Reset	Description
31:28	0x1	SYS_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	0x0	COP_AUTO_SWAKEUP_FROM_FIQ: 0 = NOP: 1 = Burst on COP FIQ
26	0x0	CPU_AUTO_SWAKEUP_FROM_FIQ: 0 = NOP: 1 = Burst on CPU FIQ
25	0x0	COP_AUTO_SWAKEUP_FROM_IRQ: 0 = NOP: 1 = Burst on COP IRQ
24	0x0	CPU_AUTO_SWAKEUP_FROM_IRQ: 0 = NOP: 1 = Burst on CPU IRQ

Bit	Reset	Description
14:12	0x0	SWAKEUP_FIQ_SOURCE: 000 = clk_m, 001 = pllC_out1, 010 = pllC4_out3, 011 = pllP_out0, 100 = pllP_out2, 101 = pllC4_out1, 110 = clk_s, 111 = pllC4_out2 0 = CLKM 1 = PLLC_OUT1 2 = PLLC4_OUT3 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC4_OUT1 6 = CLK_S 7 = PLLC4_OUT2
10:8	0x0	SWAKEUP_IRQ_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLC4_OUT3PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC4_OUT1PLLC_OUT0 6 = CLK_SCLKS 7 = PLLC4_OUT2
6:4	0x0	SWAKEUP_RUN_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLC4_OUT3PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC4_OUT1PLLC_OUT0 6 = CLK_SCLKS 7 = PLLC4_OUT2
2:0	0x0	SWAKEUP_IDLE_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLC4_OUT3PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = PLLC4_OUT1PLLC_OUT0 6 = CLK_SCLKS 7 = PLLC4_OUT2

5.2.10 CLK_RST_CONTROLLER_SUPER_SCLK_DIVIDER_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0000xxxxxxxx0000000000000000)

Bit	Reset	Description
31	DISABLE	SUPER_SDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
27	0x0	SUPER_SDIV_DIS_FROM_COP_FIQ: 0 = COP FIQ does not impact super clock divider enable, 1 = disable super clock divider on BPMP-Lite FIQ.
26	0x0	SUPER_SDIV_DIS_FROM_CPU_FIQ: 0 = CPU FIQ does not impact super clock divider enable, 1= disable super clock divider on CPU FIQ.
25	0x0	SUPER_SDIV_DIS_FROM_COP_IRQ: 0 = COP IRQ does not impact super clock divider enable, 1= disable super clock divider on BPMP-Lite IRQ.
24	0x0	SUPER_SDIV_DIS_FROM_CPU_IRQ: 0 = CPU IRQ does not impact super clock divider enable IRQ, 1 = disable super clock divider on CPU IRQ.
15:8	0x0	SUPER_SDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_SDIV_DIVISOR: Actual value = n + 1.

5.2.11 CLK_RST_CONTROLLER_CLK_SYSTEM_RATE_0

HCLK/PCLK

SCLK is the main system clock which can run up to 408 MHz.

HCLK is the AHB clock which can run at 1, 1/2, 1/3, or 1/4 of SCLK.

PCLK is the APB clock which can run at 1, 1/2, 1/3, or 1/4 of HCLK.

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x000x00)

Bit	Reset	Description
7	0x0	HCLK_DIS: 0=enable HCLK, 1=disable HCLK.
5:4	0x0	AHB_RATE: 1/(n+1) of SCLK.
3	0x0	PCLK_DIS: 0=enable PCLK, 1=disable PCLK.
1:0	0x0	APB_RATE: 1/(n+1) of HCLK.

5.2.12 CLK_RST_CONTROLLER_CLK_MASK_ARM_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

Bit	Reset	Description
16	0x0	CLK_MASK_CPU_HALT: WARNING: Deprecated. This bit must not be set in SMP mode or when the fastsync FIFO feature is enabled.
1:0	0x0	CLK_MASK_COP: 00 = no clock masking 01 = u2_nwait_r 10 = u2_nwait_r 11 = no clock masking.

5.2.13 CLK_RST_CONTROLLER_MISC_CLK_ENB_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxxx0000xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:22	0x0	DEV1_OSC_DIV_SEL: 00 = osc 01 = osc/2 10 = osc/4 11 = osc/8.
21:20	0x0	DEV2_OSC_DIV_SEL: 00 = osc 01 = osc/2 10 = osc/4 11 = osc/8.

5.2.14 CLK_RST_CONTROLLER_OSC_CTRL_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x500003f1 (0b01010000000000x00000xx111111x0x1)

Bit	Reset	Software Default	Description
31:28	OSC38P4	NONE	OSC_FREQ: 0000 = 13.0 MHz, 0100 = 19.2 MHz, 1000 = 12.0 MHz, 1100 = 26.00 MHz, 0001 = 16.8 MHz, 0101 = 38.4 MHz*, 1001 = 48.0 MHz*. *Set PLL_REF_DIV to /1 even osc = or 48 MHz. Unused code map to 38.4 MHz setting in hardware. 0 = OSC13 4 = OSC19P2 8 = OSC12 12 = OSC26 1 = OSC16P8 5 = OSC38P4 9 = OSC48
27:26	0x0	NONE	PLL_REF_DIV: PLL reference clock divide. 00 = /1 01 = /2 10 = /4 0 = DIV1 1 = DIV2 2 = DIV4 3 = RESERVED
25:18	0x0	NONE	OSCFI_SPARE: Crystal oscillator spare register control.
16:12	0x0	NONE	XODS: Crystal oscillator duty cycle control.
9:4	0x3f	0x1	XOFS: Crystal oscillator drive strength control.
2	0x0	NONE	CLK_OK: Crystal oscillator clk_ok.
0	0x1	NONE	XOE: Crystal oscillator enable (1 = enable).

5.2.15 CLK_RST_CONTROLLER_OSC_FREQ_DET_0

Oscillator Frequency Detect

The supported crystal frequency for “osc” in Tegra X1 devices is 38.4 MHz. Hardware provides the following mechanism to detect which frequency of osc is running. To determine the osc frequency, program the number of 32.768 kHz clock periods to create a fixed time window in which the "osc" clocks will be counted. Once the counting process is done, the count value can be used to determine the osc frequency.

OSC Frequency (MHz)	Approximate OSC_FREQ_DET_CNT (using two 32 kHz periods as window)
38.40	2344 (0x0928)
12.00	732 (0x02DC)
13.00	794 (0x031A)
16.80	1025 (0x0401)
19.20	1172 (0x0494)
26.00	1587 (0x0633)
48.00	2930 (0x0B72)

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	DISABLE	OSC_FREQ_DET_TRIG: 0 = default, 1 = enable osc frequency detect. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3:0	0x0	REF_CLK_WIN_CFG: Indicates the number of 32.768 kHz clock periods as window in n+1 scheme.

5.2.16 CLK_RST_CONTROLLER_OSC_FREQ_DET_STATUS_0

Offset: 0x5c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	OSC_FREQ_DET_BUSY: 0 = not busy, 1 = busy.
15:0	X	OSC_FREQ_DET_CNT: Indicates the number of osc counts within the 32.768 kHz clock reference window.

5.2.17 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x00005c00 (0b00000000000000000010111x000000000)

Bit	Reset	Software Default	Description
31:30	0x0	NONE	PLLE_INTEGOFFSET: interpolator bias current.
29:24	0x0	0x23	PLLE_SSCINCINTRV: Triangle generator increment interval control.
23:16	0x0	0x1	PLLE_SSCINC: Triangle generator increment control.
15	0x0	NONE	PLLE_SSCINVERT: 0 gives down spread, 1 gives up-spread.
14	0x1	0x0	PLLE_SSCCENTER: 0 gives control to SSCINVERT, 1 enables center spread.
13	0x0	NONE	PLLE_SSCPDMBYP: Bypass from pulse density modulator. Normally set to zero.
12	0x1	NONE	PLLE_SSCBYP: 0 enables spreading, 1 disables spreading.
11	0x1	NONE	PLLE_INTERP_RESET: Interpolator reset. 0=normal operation 1=resets SS machine.
10	0x1	NONE	PLLE_BYPASS_SS: When set feedback clock bypasses interpolator. Default value is zero.
8:0	0x0	0x21	PLLE_SSCMAX: Spread limit control.

5.2.18 CLK_RST_CONTROLLER_PLLE_MISC1_0

Offset: 0x6c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx10000)

Bit	Reset	Description
4	0x1	PLLE_SDM_RESET
3	0x0	PLLE_EN_DITHER2
2	0x0	PLLE_EN_DITHER
1	0x0	PLLE_EN_SSC
0	0x0	PLLE_EN_SDM

5.2.19 CLK_RST_CONTROLLER_PLLC_BASE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x0X108002 (0b000xxxx00001xx00100000xx00000010)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC_ENABLE: 0 = DISABLE 1 = ENABLE (will invert and connect to iddq)

Bit	R/W	Reset	Description
29	RW	REF_ENABLE	PLLC_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. To go into the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC_LOCK: 0 = not lock, 1 = lock.
26	RO	X	PLLC_LOCK: 0 = not locked, 1 = is locked.
24:20	RW	0x1	PLLC_DIVP: 0 = post divider (divide By N+1).
17:10	RW	0x20	PLLC_DIVN: PLL feedback divider.
7:0	RW	0x2	PLLC_DIVM: PLL input divider.

5.2.20 CLK_RST_CONTROLLER_PLLC_OUT_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxx000000000xxxxxx10)

Bit	Reset	Description
16	0x0	PLLC_OUT1_DIV_BYP: 1 = bypass PLLC_OUT1 divider - Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
15:8	0x0	PLLC_OUT1_RATIO: PLLC_OUT1 divider from base PLLC. Divide by [(N/2)+1].
1	ENABLE	PLLC_OUT1_CLKEN: PLLC_OUT1 divider clk enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	PLLC_OUT1_RSTN: PLLC_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.21 CLK_RST_CONTROLLER_PLLC_MISC_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x40000000 (0bx1xxxxxxxxx00000000000000000x00)

Bit	Reset	Software Default	Description
30	0x1	NONE	PLLC_RESET: Reset for digital logic of the PLL - 0 = Out of reset, 1 = Reset asserted. 0 = DISABLE 1 = ENABLE
19:4	0x0	0x8000	PLLC_EXT_FRU
3	0x0	NONE	PLLC_PTS: Base PLLC test output select. 0 = PTO is 0 1 = PTO is FO2:2 rw PLLC_EN_FRAC i=0x0 0 = DISABLE 1 = FO
1:0	0x0	NONE	PLLC_LOOP_CTRL

5.2.22 CLK_RST_CONTROLLER_PLLC_MISC_1_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1xxxxxxxxxxxxx0000xx00000000)

Bit	Reset	Description
27	0x1	PLLC_IDDQ 0 = OFF 1 = ON
13:10	0x0	PLLC_EXT_SUBINT
7:0	0x0	PLLC_DIVN_FRAC

5.2.23 CLK_RST_CONTROLLER_PLLM_BASE_0

Offset: 0x90 | Read/Write: R/W | Reset: 0x0X002a02 (0b000xxxx00000xxxx0010101000000010)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLM_BYPASSPLL: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLM_ENABLE: 0 = disable, 1 = enable. (Will invert and connect to IDDQ) 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLM_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLM_LOCK: 0 = not lock, 1 = lock. Phase and frequency
26	RO	X	PLLM_FREQ_LOCK: 0 = not lock, 1 = lock. Frequency lock only
24:20	RW	0x0	PLLM_DIVP: PLL post divider
15:8	RW	0x2a	PLLM_DIVN: PLL feedback divider.
7:0	RW	0x2	PLLM_DIVM: PLL input divider.

5.2.24 CLK_RST_CONTROLLER_PLLM_MISC1_0

Offset: 0x98 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	PLLM_SETUP: SETUP fields

5.2.25 CLK_RST_CONTROLLER_PLLM_MISC2_0

Offset: 0x9c │ Read/Write: R/W │ Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxx000000xx010000)

Bit	Reset	Description
13	0x0	PLLM_OVERRIDE_SYNCMUX: Overrides syncmux glltchless mechanism
12	0x0	PLLM_VCO_SEL: Select b/w pllma,pllmb using sync mux ctrl
11	0x0	PLLM_SYNC_MUX_CTRL: Controls clkoutp/n
10	0x0	PLLM_ENABLE_SW_OVERRIDE: Let software override values on clamp/bypass versus hardware FSM control
9:8	0x0	PLLM_PTS: PTS field for external mux. Bit 8 is for PLLM and bit 9 is for PLLMB. 0 = PTO is 0 1 = PTO is FO 0 = DISABLE 1 = FO
6	0x0	PLLM_EN_FSTLCK
5	0x0	PLLM_IDDQ
4	0x1	PLLM_EN_LCKDET:1 = Power down lock detect 0 = DISABLE 1 = ENABLE
3	0x0	PLLM_LOCK_OVERRIDE: Forces lock to 1
2:1	0x0	PLLM_KCP: Charge Pump Gain control
0	0x0	PLLM_KVCO:VCO gain

5.2.26 CLK_RST_CONTROLLER_PLLP_BASE_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0X115408 (0b0000xxx00001xx01010101xx00001000)

Bit	R/W	Reset	Software Default	Description
31	RW	DISABLE	NONE	PLLP_BYPASS: 0 = no bypass 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	ENABLE	PLLP_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	NONE	PLLP_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
28	RW	DISABLE	NONE	PLLP_BASE_OVRRIDE: 0 = disallow base override 1 = allow base override 0 = DISABLE 1 = ENABLE
27	RO	X	NONE	PLLP_LOCK: 0 = not lock, 1 = lock.
24:20	RW	0x1	NONE	PLLP_DIVP: 0 = post divider (2^n).
17:10	RW	0x55	NONE	PLLP_DIVN: PLL feedback divider.
7:0	RW	0x8	NONE	PLLP_DIVM: PLL input divider.

5.2.27 CLK_RST_CONTROLLER_PLLP_OUTA_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxx00000000xxxxx010)

Bit	Reset	Software Default	Description
15:8	0x0	NONE	PLLP_OUT1_RATIO: PLLP_OUT1 divider from base PLLP. Divide by [(N/2)+1].
2	DISABLE	NONE	PLLP_OUT1_OVRRIDE: 0 = Disallow PLLP_OUT1 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
1	ENABLE	DISABLE	PLLP_OUT1_CLKEN: PLLP_OUT1 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	NONE	PLLP_OUT1_RSTN: PLLP_OUT1 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.28 CLK_RST_CONTROLLER_PLLP_OUTB_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00020002 (0b00000000xxxxx01000000000xxxxx010)

Bit	Reset	Software Default	Description
31:24	0x0	NONE	PLLP_OUT4_RATIO: PLLP_OUT4 divider from base PLLP. Divide by [(N/2)+1].

Bit	Reset	Software Default	Description
18	DISABLE	ENABLE	PLLP_OUT4_OVRRIDE: 0 = disallow PLLP_OUT4 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
17	ENABLE	NONE	PLLP_OUT4_CLKEN: PLLP_OUT4 divider clock enable. 0 = DISABLE 1 = ENABLE
16	RESET_ENABLE	NONE	PLLP_OUT4_RSTN: PLLP_OUT4 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE
15:8	0x0	0x6	PLLP_OUT3_RATIO: PLLP_OUT3 divider from base PLLP. Divide by [(N/2)+1].
2	DISABLE	ENABLE	PLLP_OUT3_OVRRIDE: 0 = Disallow PLLP_OUT3 ratio override. 0 = DISABLE 1 = ENABLE
1	ENABLE	NONE	PLLP_OUT3_CLKEN: PLLP_OUT3 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	NONE	PLLP_OUT3_RSTN: PLLP_OUT3 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.29 CLK_RST_CONTROLLER_PLLP_MISC_0

Offset: 0xac │ Read/Write: R/W │ Reset: 0x000400X0 (0bxxxx00x000xxx10xxxxxxxxxxxxx0000)

Bit	Read/Write	Reset	Description
27	RW	0x0	PLLP_OUT4_DIV_BYP: 1 = bypass PLLP_OUT4 divider. Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
26	RW	0x0	PLLP_OUT3_DIV_BYP: 1 = bypass PLLP_OUT3 divider. Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
24	RW	0x0	PLLP_OUT1_DIV_BYP: 1 = bypass PLLP_OUT1 divider. Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
23:22	RW	0x0	PLLP_PTS: Base PLLP test output select. Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch. 00 = PTO is 0 01 = PTO is FO 10 = PTO is VCO 11 = PTO is 0 0 = DISABLE 1 = FO 2 = VCO
18	RW	0x1	PLLP_ EN_LCKDET: 0 = DISABLE 1 = ENABLE
17	RW	0x0	PLLP_LOCK_OVERRIDE: Lock override.
4	RO	X	PLLP_ FREQLOCK
3	RW	0x0	PLLP_IDDQ
2	RW	0x0	PLLP_KVCO: Base PLLP VCO range setup control.
1:0	RW	0x0	PLLP_KCP

5.2.30 CLK_RST_CONTROLLER_PLLA_BASE_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x0X803003 (0b000xxx001000xxxx0011000000000011)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLA_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLA_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLA_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLA_LOCK: 0 = not lock, 1 = lock.
26	RO	X	PLLA_FREQLOCK
25	RW	0x0	PLLA_IDDQ
24:20	RW	0x8	PLLA_DIVP: PLL post divider
15:8	RW	0x30	PLLA_DIVN: PLL feedback divider.
7:0	RW	0x3	PLLA_DIVM: PLL input divider.

5.2.31 CLK_RST_CONTROLLER_PLLA_OUT_0

See the “PLL post dividers pll*_out*” section.

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxx00000000xxxxxx10)

Bit	Reset	Description
15:8	0x0	PLLA_OUT0_RATIO: PLLA_OUT0 divider from base PLLA. Divide by [(N/2)+1]. 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLA_OUT0_CLKEN: PLLA_OUT0 divider clk enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	PLLA_OUT0_RSTN: PLLA_OUT0 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.32 CLK_RST_CONTROLLER_PLLA_MISC1_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLA_SDM_DIN_NEW
15:0	0x0	PLLA_SDM_DIN

5.2.33 CLK_RST_CONTROLLER_PLLA_MISC_0

Offset: 0xbc | Read/Write: R/W | Reset: 0x12000000 (0bx00100100000000000000000000000000)

Bit	Reset	Software Default	Description
30	0x0	NONE	PLLA_OUT0_DIV_BYP: 1 = bypass PLLA_OUT0 divider. Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
29	0x0	NONE	PLLA_PTS: Base PLLA test output select. 0 = PTO is 0 1 = PTO is FO 0 = DISABLE 1 = FO

Bit	Reset	Software Default	Description
28	0x1	NONE	PLLA_EN_LCKDET: 0 = DISABLE 1 = ENABLE
27	0x0	NONE	PLLA_LOCK_OVERRIDE: lock override.
26:25	0x1	NONE	PLLA_KCP: Base PLLA charge pump setup control.
24	0x0	NONE	PLLA_KVCO: Base PLLA VCO range setup control.
23:0	0x0	0x20	PLLA_SETUP: PLLA setup control

5.2.34 CLK_RST_CONTROLLER_PLLU_BASE_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x0X011902 (0b000xxx01000000010001100100000010)

Bit	R/W	Reset	Software Default	Description
31	RW	DISABLE	NONE	PLLU_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	NONE	PLLU_ENABLE: This bit is used only when the PLLU_OVERRIDE bit is set. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	NONE	PLLU_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	NONE	PLLU_LOCK: 0 = not lock, 1 = lock.
25	RW	0x0	NONE	PLLU_CLKENABLE_48M: FO_48M output enable. This bit is use only when the PLLU_OVERRIDE bit is set.
24	RW	0x1	0x0	PLLU_OVERRIDE: 0 = FO_[ICUSB,HSIC,USB] controlled by USB controllers, 1 = controlled by PLLU_CLKENABLEs.
23	RW	0x0	NONE	PLLU_CLKENABLE_ICUSB: FO_ICUSB output enable. This bit is used only when the PLLU_OVERRIDE bit is set.
22	RW	0x0	NONE	PLLU_CLKENABLE_HSIC: FO_HSIC output enable. This bit is used only when the PLLU_OVERRIDE bit is set. Otherwise, USB controllers will control this automatically. 0 = disable 1 = enable
21	RW	0x0	NONE	PLLU_CLKENABLE_USB: FO_USB output enable. This bit is used only when the PLLU_OVERRIDE bit is set. Otherwise, USB controllers will control this automatically. 0 = disable 1 = enable
20:16	RW	0x1	NONE	PLLU_ DIVP: PL divider
15:8	RW	0x19	NONE	PLLU_DIVN: PLL feedback divider.
7:0	RW	0x2	NONE	PLLU_DIVM: PLL input divider.

5.2.35 CLK_RST_CONTROLLER_PLLU_OUTA_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00020002 (0b00000000xxxxx01000000000xxxxx010)

Bit	Reset	Software Default	Description
31:24	0x0	0x6	PLLU_OUT2_RATIO: PLLU_OUT2 divider from base PLLU - Divide by [(N/2)+1].
18	DISABLE	ENABLE	PLLU_OUT2_OVRRIDE: 0 = disallow PLLU_OUT2 ratio override, 1 = enable override. 0 = DISABLE 1 = ENABLE

Bit	Reset	Software Default	Description
17	ENABLE	NONE	PLLU_OUT2_CLKEN: PLLU_OUT2 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
16	RESET_ENABLE	NONE	PLLU_OUT2_RSTN: PLLU_OUT2 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE
15:8	0x0	0x8	PLLU_OUT1_RATIO: PLLU_OUT1 divider from base PLLU - Divide by [(N/2)+1].
2	DISABLE	ENABLE	PLLU_OUT1_OVRRIDE: 0 = disallow PLLU_OUT1 ratio override, 1 = enable override. 0 = DISABLE 1 = ENABLE
1	ENABLE	NONE	PLLU_OUT1_CLKEN: PLLU_OUT1 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	NONE	PLLU_OUT1_RSTN: PLLU_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.36 CLK_RST_CONTROLLER_PLLU_MISC1_0

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	PLLU_OUT2_DIV_BYP: 1 = bypass PLLU_OUT2 divider - Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
1	0x0	PLLU_OUT1_DIV_BYP: 1 = bypass PLLU_OUT1 divider - Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
0	0x0	PLLU_LOCK_OVERRIDE: lock override.

5.2.37 CLK_RST_CONTROLLER_PLLU_MISC_0

Offset: 0xcc | Read/Write: R/W | Reset: 0xX0000000 (0b0x10000000000000000000000000000)

Bit	Read/Write	Reset	Description
31	RW	0x0	PLLU_IDDQ: 0 = OFF 1 = ON
30	RO	X	PLLU_FREQLOCK: 0 = not lock, 1 = lock frequency
29	RW	0x1	PLLU_EN_LCKDET: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
28:27	RW	0x0	PLLU_PTS: Base PLLU test output select. 00 = PTO is 0 01 = PTO is VCO 10 = PTO is FO 11 = PTO is FO_ICUSB 0 = DISABLE 1 = VCO 2 = FO 3 = FO_ICUSB
26:25	RW	0x0	PLLU_KCP: Base PLLU charge pump setup control.
24	RW	0x0	PLLU_KVCO: Base PLLU VCO gain settings
23:0		0x0	PLLU_SETUP: Base PLLU setup control.

5.2.38 CLK_RST_CONTROLLER_PLLD_BASE_0

Offset: 0xd0 | Read/Write: R/W | Reset: 0x0X211001 (0b000xxx0x0010x00100010xxx00000001)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD_REF_DIS: 0 = Enable reference clock, 1 = Disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLD_LOCK: 0 = not lock, 1 = lock.
25	RW	PLL_D	DSIA_CLK_SRC: Only PLLD is the source because it is only MIPI PLL. 0 = PLL_D 1 = PLL_D1
23	RW	0x0	CSI_CLK_SRC: Selects the source of the CSI clock. 0 = brick: typically selected when using an external sensor that provides its own clock 1 = PLL_D: must be selected when the TPG (test pattern generator) is used
22:20	RW	0x2	PLLD_DIVP: 0 = Post divider (2^n).
18:11	RW	0x22	PLLD_DIVN: PLL feedback divider.
7:0	RW	0x1	PLLD_DIVM: PLL input divider.

5.2.39 CLK_RST_CONTROLLER_PLLD_MISC1_0

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Software Default	Description
23:0	0x0	0x20	PLLD_SETUP

5.2.40 CLK_RST_CONTROLLER_PLLD_MISC_0

Offset: 0xdc | Read/Write: R/W | Reset: 0x000X0000 (0bxx0000000000x10000000000000000000)

Bit	Read/Write	Reset	Description
29:27	RW	0x0	PLLD_LDPULSE_ADJ: load pulse position adjust.
26:25	RW	0x0	PLLD_PTS: Base PLLD test output select. 00 = PTO is 0 01 = PTO is FO 10 = PTO is 0 11 = PTO is 0 0 = DISABLE 1 = FO
24:23	RW	0x0	PLLD_KCP: Base PLLD charge pump setup control
22	RW	0x0	PLLD_KVCO: Base PLLD VCO range setup control.
21	RW	0x0	PLLD_ENABLE_CLK: Controls the differential clock from PLLD to the DSI and CSI pads. 0 = disable 1 = enable
20		0x0	PLLD_IDDQ
19	RO	X	PLLD_FREQLOCK
18	RW	0x1	PLLD_EN_LCKDET
17	RW	0x0	PLLD_LOCK_OVERRIDE: lock select
16	RW	0x0	PLLD_EN_SDM
15:0	RW	0x0	PLLD_SDM_DIN

5.2.41 CLK_RST_CONTROLLER_PLLX_BASE_0

Offset: 0xe0 | Read/Write: R/W | Reset: 0x0X002401 (0b000xxxx00000xxxx0010010000000001)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLX_BYPASS: Reserved
30	RW	DISABLE	PLLX_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLX_REF_DIS: 0 = Enable reference clock, 1 = Disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLX_LOCK: 0 = Not lock, 1 = Lock.
24:20	RW	0x0	PLLX_DIVP: 0 = Post divider (2^n).
15:8	RW	0x24	PLLX_DIVN: PLL feedback divider.
7:0	RW	0x1	PLLX_DIVM: PLL input divider.

5.2.42 CLK_RST_CONTROLLER_PLLX_MISC_0

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxxx00xxx0xxxxxxxxxxxxxxxxxx)

Bit	Reset	Software Default	Description
28	0x0	NONE	PLLX_FO_G_DISABLE: PLLX FO_G output disable. 0=ON, 1=OFF
23:22	0x0	NONE	PLLX_PTS: Base PLLX test output select. 00 = PTO is 0 01 = PTO is FO 10 = PTO is VCO 11 = PTO is 0 0 = DISABLE 1 = FO 2 = VCO
18	0x0	0x1	PLLX_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE

5.2.43 CLK_RST_CONTROLLER_PLLE_BASE_0

Offset: 0xe8 | Read/Write: R/W | Reset: 0x0e007d02 (0b00001110000000000111110100000010)

Bit	Reset	Description
31	DISABLE	PLLE_ENABLE: PLL enable. 0 = DISABLE 1 = ENABLE
30	0x0	PLLE_LOCK_OVERRIDE: Forces PLL_LOCK and PLL_FREQLOCK to 1.
29	0x0	PLLE_FDIV4B: 0 = vcoclk/4, 1 = vcoclk/2 clock to the interpolator logic. Normally set to 0.
28:24	0xe	PLLE_PLDIV_CML: Divider control for CLOCKOUT_CML/CLOCKOUTB_CML.
23:16	0x0	PLLE_EXT_SETUP_23_16: Base PLLE setup [19:16].
15:8	0x7d	PLLE_NDIV: Feedback divider.
7:0	0x2	PLLE_MDIV: Input divider.

5.2.44 CLK_RST_CONTROLLER_PLLE_MISC_0

Offset: 0xec | Read/Write: R/W | Reset: 0x0000XX00 (0b00000000000000000011xx010000000x0)

Bit	R/W	Reset	Description
31:16	RW	0x0	PLLE_SETUP: Base PLLE setup [15:0].
15	RW	0x0	PLLE_CLKENABLE: Clock gate enable for PLLE output going to mccenter.

Bit	R/W	Reset	Description
14	RW	0x1	PLLE_IDDQ_SWCTL: 0 = The PLLE is put in IDDQ mode by hardware (SATA +AFI+CAR) signals.1 = The PLLE is put in IDDQ by software. 0 = OFF 1 = ON (default)
13	RW	0x1	PLLE_IDDQ_OVERRIDE_VALUE: 0 = The PLLE is powered up. 1 = Software can put the PLLE in IDDQ by setting this bit and PLLE_IDDQ_SWCTL 0 = OFF 1 = ON (default)
12	RO	X	PLLE_FREQLOCK: 0 = frequency acquisition not achieved, 1 = frequency acquisition occurred. PLLE_LOCK_ENABLE must be enabled.
11	RO	X	PLLE_LOCK: 0 = not lock (phase+frequency), 1 = lock (phase+frequency). PLLE_LOCK_ENABLE must be enabled.
10	RW	REF_ENABLE	PLLE_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
9	RW	0x1	PLLE_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
8	RW	0x0	PLLE_PTS: ~PLLE_BYPASS - PLLE_PTS needs to be set to 1 for the output to be from PLLE. PLLE_PTS as 0 implies that the reference clock is sent and PLLE is bypassed. This needs to be set before enabling PLLE. 0 = PTO is always 0 if PLLE_ENABLE=0 (SYN_CLOCKOUT=0) 0 = PTO = PLLE CLOCKIN if PLLE_ENABLE=1 1 = PTO = PLLE FO (SYN_CLOCKOUT=VCOCLOCK/PLDIV).
7:6	RW	0x0	PLLE_KCP: Base PLLE charge pump gain control.
5:4	RW	0x0	PLLE_VREG_BG_CTRL: Base PLLE VREG BG control.
3:2	RW	0x0	PLLE_VREG_CTRL: Base PLLE VREG control.
0	RW	0x0	PLLE_KVCO: Base PLLE VCO gain.

5.2.45 CLK_RST_CONTROLLER_PLLE_SS_CNTL1_0

Offset: 0xf0 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLE_SDM_DIN
15:0	0x0	PLLE_SDM_SSC_STEP

5.2.46 CLK_RST_CONTROLLER_PLLE_SS_CNTL2_0

Offset: 0xf4 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLE_SDM_SSC_MAX
15:0	0x0	PLLE_SDM_SSC_MIN

5.2.47 CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRA_0

Offset: 0xf8 I Read/Write: R/W I Reset: 0x00000000 (0bx0x0xxxxxxxxxxx0x00x00000xx000x)

Bit	Reset	Description
30	0x0	PPCS_CLK_OVR_ON
28	0x0	BSEV_CLK_OVR_ON
15	0x0	VI_CLK_OVR_ON
13	0x0	MPCORE_CLK_OVR_ON
12	0x0	MC_CLK_OVR_ON
10	0x0	HC_RIF_CLK_OVR_ON
9	0x0	HC_RDMA_CLK_OVR_ON
8	0x0	HC_INTFC_OVR_ON
7	0x0	HC_CLK_OVR_ON

Bit	Reset	Description
6	0x0	HC_CDMA_CLK_OVR_ON
3	0x0	EMC_CLK_OVR_ON
2	0x0	DCB_CLK_OVR_ON
1	0x0	DC_CLK_OVR_ON

5.2.48 CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRB_0

Offset: 0xfc I Read/Write: R/W I Reset: 0x00000000 (0bx0x00xxxxxxxxxxxx0x00xxxxxxxx0x)

Bit	Reset	Description
30	0x0	SE_CLK_OVR_ON
28	0x0	AFI_CLK_OVR_ON
27	0x0	MPCORELP_CLK_OVR_ON: These bits are no longer used, but to maintain Software compatibility they may not be recycled. 26:26 rw GR3D2_VPECLK_OVR_ON i=0x0 25:25 rw GR3D2_TEXCLK_OVR_ON i=0x0 24:24 rw GR3D2_SETUPCLK_OVR_ON i=0x0 23:23 rw GR3D2_QRASTCLK_OVR_ON i=0x0 22:22 rw GR3D2_PSEQCLK_OVR_ON i=0x0 21:21 rw GR3D2_IDXCLK_OVR_ON i=0x0 20:20 rw GR3D2_FDCCLK_OVR_ON i=0x0 19:19 rw GR3D2_DWRCLK_OVR_ON i=0x0 18:18 rw GR3D2_CLIPCLK_OVR_ON i=0x0 17:17 rw GR3D2_ATRASTCLK_OVR_ON i=0x0 16:16 rw GR3D2_ALUCLK_OVR_ON i=0x0
13	0x0	USB1_CLK_OVR_ON
11	0x0	AVPC_CLK_OVR_ON
10	0x0	USB2_CLK_OVR_ON
1	0x0	CSI_CLK_OVR_ON

5.2.49 CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRC_0

Offset: 0x3a0 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000xx0000x0000000000)

Bit	Reset	Description
31	0x0	XUSB_DEV_CLK_OVR_ON
30	0x0	XUSB_HOST_CLK_OVR_ON
29	0x0	RESERVED
28	0x0	TMS0_GRPCLK_CLK_OVR_ON
27	0x0	SATA_12M_CLK_OVR_ON
26	0x0	SATA_UFPCI_CLK_OVR_ON
25	0x0	SATA_DFPCI_CLK_OVR_ON
24	0x0	SATA_CH1_RX_CLK_OVR_ON
23	0x0	SATA_CH1_TX_CLK_OVR_ON
22	0x0	SATA_TX_CLK_OVR_ON
21	0x0	SATA_FPCI_CH1_CLK_OVR_ON
20	0x0	SATA0_FPCI_CLK_OVR_ON
19	0x0	SATA_FPCI_CLK_OVR_ON
18	0x0	SATA_FPCI_ALWAYS_CLK_OVR_ON
17	0x0	SATA_IPFS_CLK_OVR_ON
14	0x0	SPI4_CLK_OVR_ON
13	0x0	SPI3_CLK_OVR_ON
12	0x0	SPI2_CLK_OVR_ON
11	0x0	SPI1_CLK_OVR_ON
9	0x0	MPCORE_MCCLK_OVR_ON: Override clock-enables on the memory-clients for the G CPU. This trades off power for better L2-miss latency.
8	0x0	TMS0_XCLK_CLK_OVR_ON
7	0x0	TMCS0_XTXCLK1X_CLK_OVR_ON
6	0x0	SDMMC4_CLK_OVR_ON
5	0x0	SDMMC3_CLK_OVR_ON
4	0x0	SDMMC2_CLK_OVR_ON
3	0x0	SDMMC1_CLK_OVR_ON
2	0x0	HDA_CLK_OVR_ON
1	0x0	AHUB_CLK_OVR_ON
0	0x0	SATA_CLK_OVR_ON

5.2.50 CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRD_0

Offset: 0x3a4 I Read/Write: R/W I Reset: 0x00000000 (0b000000x000000xxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	SDMMC4_LEGACY_TMCLK_OVR_ON
30	0x0	SDMMC3_LEGACY_TMCLK_OVR_ON
29	0x0	SDMMC2_LEGACY_TMCLK_OVR_ON
28	0x0	SDMMC1_LEGACY_TMCLK_OVR_ON
27	0x0	MPCORE_MSELECT_CLK_OVR_ON: Override clock gating to the CPU cluster MSelect clock
26	0x0	A9AVP_CLK_OVR_ON
24	0x0	QSPI_CLK_OVR_ON
23	0x0	TZRAM_CLK_OVR_ON
22	0x0	ISPB_CLK_OVR_ON
21	0x0	TSECB_CLK_OVR_ON: On the MCCIF memory clients
20	0x0	TSEC_CLK_OVR_ON: On the MCCIF memory clients
19	0x0	ARC_CLK_OVR_ON

5.2.51 CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRE_0

Offset: 0x554 I Read/Write: R/ I Reset: 0x00000000 (0b0x0xxxxxxxxxxxxxxx0000000x0x0000)

Bit	Reset	Description
31	0x0	NVDEC_CLK_OVR_ON:30:30 rw NVENC_MEDMA_CLK_OVR_ON i=0x0
29	0x0	NVENC_CLK_OVR_ON:27:27 rw NVENC_WMVP_CLK_OVR_ON i=0x026:26 rw NVENC_RMVP_CLK_OVR_ON i=0x025:25 rw NVENC_RHINT_CLK_OVR_ON i=0x024:24 rw NVENC_RCOL_CLK_OVR_ON i=0x023:23 rw NVENC_PDMA_CLK_OVR_ON i=0x022:22 rw NVENC_MPEC_CLK_OVR_ON i=0x021:21 rw NVENC_MPEB_CLK_OVR_ON i=0x020:20 rw NVENC_ME_CLK_OVR_ON i=0x019:19 rw NVENC_MDP_CLK_OVR_ON i=0x018:18 rw NVENC_HIST_CLK_OVR_ON i=0x017:17 rw NVENC_HHREG_CLK_OVR_ON i=0x0
13	0x0	ETR_CLK_OVR_ON
12	0x0	AXIAP_CLK_OVR_ON
11	0x0	ADSP_CLK_OVR_ON
10	0x0	APE_CLK_OVR_ON
9	0x0	NVJPG_CLK_OVR_ON
8	0x0	SOR1_CLK_OVR_ON
7	0x0	SOR0_CLK_OVR_ON
5	0x0	VIC_CLK_OVR_ON
3	0x0	ISP_CLK_OVR_ON
2	0x0	SATA_CH2_RX_CLK_OVR_ON
1	0x0	SATA_CH2_TX_CLK_OVR_ON
0	0x0	SATA_FPCI_CH2_CLK_OVR_ON

5.2.52 CLK_RST_CONTROLLER_CLK_SOURCE_I2S2_0

Offset: 0x100 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	I2S2_CLK_SRC: 000 = pllA_out0 010 = audio SYNC_CLK 1x or 2x 100 = pllP_out0 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S2_MASTER_CLKEN: Reserved.
7:0	0x0	I2S2_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.53 CLK_RST_CONTROLLER_CLK_SOURCE_I2S3_0

Offset: 0x104 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	I2S3_CLK_SRC: 000 = pllA_out0 010 = audio SYNC_CLK 1x or 2x 100 = pllP_out0 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S3_MASTER_CLKEN: Reserved.
7:0	0x0	I2S3_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.54 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_OUT_0

Offset: 0x108 │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPDIFOUT_CLK_SRC: 000 = pllA_out0 010 = audio SYNC_CLK 1x or 2x 100 = pllP_out0 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
7:0	0x0	SPDIF_OUT_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.55 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_IN_0

Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	SPDIF_IN_CLK_SRC: 000 = pllP_out0 001 = pllC2_out0 010 = pllC_out0 011 = pllC4_out0 101 = clk_m 110 = pllC4_out1 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = CLK_M 6 = PLLC4_OUT1 7 = PLLC4_OUT2
7:0	0x0	SPDIF_IN_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.56 CLK_RST_CONTROLLER_CLK_SOURCE_PWM_0

Offset: 0x110 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	PWM_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 100 = clk_s, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = CLK_S 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	PWM_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.57 CLK_RST_CONTROLLER_CLK_SOURCE_SPI2_0

Offset: 0x118 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPI2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	SPI2_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.58 CLK_RST_CONTROLLER_CLK_SOURCE_SPI3_0

Offset: 0x11c | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPI3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	SPI3_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.59 CLK_RST_CONTROLLER_CLK_SOURCE_I2C1_0

Offset: 0x124 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
15:0	0x0	I2C1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

5.2.60 CLK_RST_CONTROLLER_CLK_SOURCE_ I2C5_0

Offset: 0x128 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C5_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
15:0	0x0	I2C5_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

5.2.61 CLK_RST_CONTROLLER_CLK_SOURCE_SPI1_0

Offset: 0x134 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPI1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	SPI1_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.62 CLK_RST_CONTROLLER_CLK_SOURCE_DISP1_0

Offset: 0x138 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	CLK_M	DISP1_CLK_SRC: 000 = pllP_out0, 001 = pllD_out, 010 = pllD_out0, 110 = clk_m, 101 = pllD2_out0. Non sequential mapping used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 1 = PLLD_OUT 2 = PLLD_OUT0 6 = CLK_M 5 = PLLD2_OUT0

5.2.63 CLK_RST_CONTROLLER_CLK_SOURCE_DISP2_0

Offset: 0x13c | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	CLK_M	DISP2_CLK_SRC: 000 = pllP_out0, 010 = pllD_out0, 101 = pllD2_out0, 110 = clk_m, 111 = 1'b0. Non sequential mapping used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 2 = PLLD_OUT0 5 = PLLD2_OUT0 6 = CLK_M

5.2.64 CLK_RST_CONTROLLER_CLK_SOURCE_ISP_0

Offset: 0x144 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	ISP_CLK_SRC: 001 = pllC_out0, 010 = pllP_out0, 011 = pllA1_out0, 100 = pllC2_out0, 101 = pllC3_out0, 110 = clk_m, 111 = pllC4_out0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA1_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	ISP_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.65 CLK_RST_CONTROLLER_CLK_SOURCE_VI_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x80000000 (0b100xxx00xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	VI_CLK_SRC: 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 101 = CLK_M, 110 = pllA1_out0, 111 = pllC4_out0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 5 = CLK_M 6 = PLLA1_OUT0 7 = PLLC4_OUT0
25	0x0	PD2VI_CLK_SEL: 0 = pd2vi_clk, 1 = vi_sensor_clk.
24	INTERNAL	VI_CLK_SEL: 0 = select internal clock, 1 = select external clock (pd2vi_clk). 0 = INTERNAL 1 = EXTERNAL
7:0	0x0	VI_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.66 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0

Offset: 0x150 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SDMMC1_CLK_SRC: 000 = pllP_out0, 001 = pllA_out, 010 = pllC_out0, 011 = pllC4_out2, 100 = pllM_out0, 101 = pllE_out0, 110 = clk_m, 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLA_OUT 2 = PLLC_OUT0 3 = PLLC4_OUT2 4 = PLLM_OUT0 5 = PLLE_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	SDMMC1_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.67 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0

Offset: 0x154 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SDMMC2_CLK_SRC: 000 = pllP_out0, 001 = pllC4_out2_LJ, 010 = pllC4_out0_LJ, 011 = pllC4_out2, 100 = pllC4_out1, 101 = pllC4_out1_LJ, 110 = clk_m, 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLC4_OUT2_LJ 2 = PLLC4_OUT0_LJ 3 = PLLC4_OUT2 4 = PLLC4_OUT1 5 = PLLC4_OUT1_LJ 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	SDMMC2_CLK_DIVISOR: N = Divide by [(N/2)+1] Note: The divisor is ignored if the *_LJ options are chosen in corresponding _SRC fields.

5.2.68 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0

Offset: 0x164 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SDMMC4_CLK_SRC: 000 = pllP_out0, 001 = pllC4_out2_LJ, 010 = pllC4_out0_LJ, 011 = pllC4_out2, 100 = pllC4_out1, 101 = pllC4_out1_LJ, 110 = clk_m, 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLC4_OUT2_LJ 2 = PLLC4_OUT0_LJ 3 = PLLC4_OUT2 4 = PLLC4_OUT1 5 = PLLC4_OUT1_LJ 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	SDMMC4_CLK_DIVISOR: N = Divide by [(N/2)+1] Note: The divisor is ignored if the *_LJ options are chosen in corresponding _SRC fields.

5.2.69 CLK_RST_CONTROLLER_CLK_SOURCE_UARTA_0

Offset: 0x178 | Read/Write: R/W | Reset: 0xc0000002 (0b110xxxx0xxxxxxxx0000000000000010)

Bit	Reset	Description
31:29	CLK_M	UARTA_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = PLLC4_OUT2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
24	DISABLE	UARTA_DIV_ENB: 0=Use UART DLM/DLL, 1=Use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTA_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.70 CLK_RST_CONTROLLER_CLK_SOURCE_UARTB_0

Offset: 0x17c | Read/Write: R/W | Reset: 0xc0000002 (0b110xxxx0xxxxxxxx0000000000000010)

Bit	Reset	Description
31:29	CLK_M	UARTB_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = PLLC4_OUT2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
24	DISABLE	UARTB_DIV_ENB: 0=Use UART DLM/DLL, 1=Use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTB_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.71 CLK_RST_CONTROLLER_CLK_SOURCE_HOST1X_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x80000000 (0b100xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	PLLP_OUT0	HOST1X_CLK_SRC: 000 = pllC4_out1 001 = pllC2_out0 010 = pllC_out0 011 = pllC4_out2 100 = pllP_out0 101 = clk_m 110 = pllA_out0 111 = pllC4_out0 0 = PLLC4_OUT1 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT2 4 = PLLP_OUT0 5 = CLK_M 6 = PLLA_OUT0 7 = PLLC4_OUT0
15:8	0x0	HOST1X_IDLE_DIVISOR: Divide by [(N/2)+1]. If all 0s, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of HOST1X_CLK_DIVISOR.
7:0	0x0	HOST1X_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.72 CLK_RST_CONTROLLER_CLK_SOURCE_I2C2_0

Offset: 0x198 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
15:0	0x0	I2C2_CLK_DIVISOR: Divide by (n+1). (lsb denotes 1.0x)

5.2.73 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_0

When changing the EMC frequency, internal logic sequencing handles the requirement that the DRAMs must be placed into self-refresh before its clock frequency is changed.

To support this, certain register fields when written do not get immediately applied but rather are applied at the correct time during the sequencing. These registers are referred to as "shadowed".

Other fields initiate the sequencing state machine and therefore should be written last. Note that the value of the field must change to start the state machine - writing the same value to the field is insufficient.

At the end of the sequencing, the DRAMs are restored to being operational.

There are EMC/MC configuration registers that are shadowed or can initiate a clock change sequence are described below:

- Initiates sequencing (value must change):
  - CLK_SOURCE_EMC.EMC_2X_CLK_SRC
  - EMC_2X_CLK_DIVISOR
  - MC_EMC_SAME_FREQ
- Shadowed (will be applied during sequencing):
  - CLK_SOURCE_EMC.EMC_2X_CLK_SRC
  - EMC_2X_CLK_DIVISOR
  - MC_EMC_SAME_FREQ
- Not shadowed:
  - CLK_SOURCE_EMC.EMC_INVERT_DCD (must be set during initial configuration and before the EMC clock is started)

Offset: 0x19c | Read/Write: R/W | Reset: 0x60180000 (0b011x000xxxx11xx00xxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_2X_CLK_SRC: 000 = pllM_out0: Clock path will be: PLLMA -> Switch -> Divider -> PLLM Bypass -> PLLM SYNC_CLKOUTP/N, 001 = pllC_out0, 010 = pllP_out0, 011 = clk_m, 100 = pllM_out_for_emc: Clock path will be: PLLMA -> PLLM SYNC_CLKOUTP/N, 101 = pllMb_out_for_emc: Clock path will be: PLLMB -> PLLM SYNC_CLKOUTP/N, 110 = pllMb_out0: Clock path will be: PLLMB -> Switch -> Divider -> PLLM Bypass -> PLLM SYNC_CLKOUTP/N, 111 = PLLP_UD. 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLMB_UD 6 = PLLMB_OUT0 7 = PLLP_UD

Bit	Reset	Description
27	0x0	FORCE_CC_TRIGGER: 1 = force a trigger of clock change sequence even if EMC_2X_CLK_SRC/EMC_2X_CLK_DIVISOR fields are unchanged.
26	DISABLE	EMC_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD. 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
25	0x0	USE_32KHZ_AS_CLK_M: 1 = CLK_M = 32kHz clock; 0 = CLK_M = OSC = clk_m
20	ENABLE	PLLC_OUT_FOR_EMC_EN: 0 = PLLC branch for EMC CLK switches disabled 1 = PLLC branch for EMC CLK switches enabled 0 = DISABLE 1 = ENABLE
19	ENABLE	PLLP_OUT_FOR_EMC_EN: 0 = PLLP branch for EMC CLK switches disabled 1 = PLLP branch for EMC CLK switches enabled 0 = DISABLE 1 = ENABLE
16	DISABLE	MC_EMC_SAME_FREQ: 0 = MC is 1/2 the frequency of EMC. 1 = MC has the same frequency as the EMC. 0 = DISABLE 1 = ENABLE
15	DISABLE	EMC_CLK_DIV2_EN: 0 = EMCCLK has the same frequency as DRAM clk 1 = EMCCLK is the 1/2 frequency of DRAM clk (LPDDR4 Mode) 0 = DISABLE 1 = ENABLE
7:0	0x0	EMC_2X_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.74 CLK_RST_CONTROLLER_CLK_SOURCE_UARTC_0

Offset: 0x1a0 | Read/Write: R/W | Reset: 0xc0000002 (0b110xxxx0xxxxxxxx0000000000000010)

Bit	Reset	Description
31:29	CLK_M	UARTC_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = PLLC4_OUT1, 110 = CLK_M, 111 = PLLC4_OUT2. 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
24	DISABLE	UARTC_DIV_ENB: 0=Use UART DLM/DLL, 1=Use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTC_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.75 CLK_RST_CONTROLLER_CLK_SOURCE_VI_SENSOR_0

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x80000000 (0b100xxxxxxxxxxxxxxxxxxxxx000000000x80000000 (0b100xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	VI_SENSOR_CLK_SRC: 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	VI_SENSOR_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.76 CLK_RST_CONTROLLER_CLK_SOURCE_SPI4_0

Offset: 0x1b4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SPI4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	SPI4_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.77 CLK_RST_CONTROLLER_CLK_SOURCE_I2C3_0

Offset: 0x1b8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
15:0	0x0	I2C3_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 1.0x)

5.2.78 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SDMMC3_CLK_SRC: 000 = pllP_out0, 001 = pllA_out, 010 = pllC_out0, 011 = pllC4_out2, 100 = pllC4_out1, 101 = pllE_out0, 110 = clk_m, 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLA_OUT 2 = PLLC_OUT0 3 = PLLC4_OUT2 4 = PLLC4_OUT1 5 = PLLE_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	SDMMC3_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.79 CLK_RST_CONTROLLER_CLK_SOURCE_UARTD_0

Offset: 0x1c0 | Read/Write: R/W | Reset: 0xc0000002 (0b110xxxx0xxxxxxxx0000000000000010)

Bit	Reset	Description
31:29	CLK_M	UARTD_CLK_SRC: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
24	DISABLE	UARTD_DIV_ENB: 0=Use UART DLM/DLL, 1=Use divisor below. 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTD_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.80 CLK_RST_CONTROLLER_CLK_SOURCE_OWR_0

This register is deprecated.

Offset: 0x1cc | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	Unused, reserved.
7:0	0x0	Unused, reserved.

5.2.81 CLK_RST_CONTROLLER_CLK_SOURCE_CSITE_0

Offset: 0x1d4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	CSITE_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllrefe_out1, 101 = pllA1_out0, 110 = clk_m, 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLREFE_OUT1 5 = PLLA1_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	CSITE_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.82 CLK_RST_CONTROLLER_CLK_SOURCE_I2S1_0

Offset: 0x1d8 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	I2S1_CLK_SRC: 000 = pllA_out0, 010 = audio SYNC_CLK 1x or 2x, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S1_MASTER_CLKEN: Reserved.
7:0	0x0	I2S1_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.83 CLK_RST_CONTROLLER_CLK_SOURCE_DTV_0

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	0x0	DTV_INV_CLK: 1 = invert DTV clock.

5.2.84 CLK_RST_CONTROLLER_CLK_SOURCE_TSEC_0

Offset: 0x1f4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	TSEC_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 101 = pllA1_out0, 110 = CLK_M (osc), 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 5 = PLLA1_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	TSEC_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.85 CLK_RST_CONTROLLER_CLK_SOURCE_LA_0

Offset: 0x1f8 | Read/Write: R/W | Reset: 0xc0000000 0b110xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	A_CLK_SRC: 000 = pllP_out0 001 = pllC2_out0 010 = pllC_out0 011 = pllC3_out0 100 = pllREFE_out1 101 = PLLA1_OUT0 110 = CLK_M 111 = PLLC4_OUT0 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLREFE_OUT1 5 = PLLA1_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	LA_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.86 CLK_RST_CONTROLLER_CLK_SPARE2_0

Offset: 0x1fc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OSC_CLK_SRC: RESERVED_CLK_SOURCE_OSC.

5.2.87 CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0

Offset: 0x280 | Read/Write: R/W | Reset: 0x23000780 (0bxx100011000xx00xx00x011110000000)

Bit	Reset	Description
29	ENABLE	CLK_ENB_PLLG_REF: Clock gate for reference clock branch going to PLLG. 0 = DISABLE 1 = ENABLE
28	0x0	CLK_ENB_PLLA_ADSP: Clock gate for PLLA branch going to ADSP. 0 = DISABLE 1 = ENABLE
27	0x0	CLK_ENB_PLLP_ADSP: Clock gate for PLLP branch going to ADSP. 0 = DISABLE 1 = ENABLE
26	0x0	CLK_ENB_HPLL_ADSP: Clock gate for HPLL(PLLC/PLLC2/PLLC3) branches going to ADSP. 0 = DISABLE 1 = ENABLE
25	ENABLE	CLK_ENB_DBGAPB: Enable clock - DBGAPB 0 = DISABLE 1 = ENABLE
24	ENABLE	CLK_ENB_GPU: Enable clock - GPU 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_SOR1: Enable clock - SOR1 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_SOR0: Enable clock - SOR0 0 = DISABLE 1 = ENABLE
21	DISABLE	CLK_ENB_DPAUX: Enable clock - DPAUX 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_VIC: Enable clock - VIC 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_UART_FST_MIPI_CAL: Enable clock - UART_FST_MIPI_CAL 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_EMC_DLL: Enable clock - EMC DLL 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_MIPIBIF: Reserved
11	DISABLE	CLK_ENB_VIM2_CLK: Enable clock - CAMERA ifc 2 0 = DISABLE 1 = ENABLE
10	ENABLE	CLK_ENB_MC_BBC: Enable clock - MC BBC 0 = DISABLE 1 = ENABLE
9	ENABLE	CLK_ENB_MC_CPU: Enable clock - MC CPU 0 = DISABLE 1 = ENABLE
8	ENABLE	CLK_ENB_MC_CBPA: Enable clock - MC daisy chain2 0 = DISABLE 1 = ENABLE
7	ENABLE	CLK_ENB_MC_CAPA: Enable clock - MC daisy chain1 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_I2C6: Enable clock - I2C6 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_CAM_MCLK2: Enable clock - CAM_MCLK2 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	DISABLE	CLK_ENB_CAM_MCLK: Enable clock - CAM_MCLK 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_ETR: Enable ETR clk. 0 = DISABLE 1 = ENABLE
2	DISABLE	CLK_ENB_DMIC2: Enable clock - DMIC2 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_DMIC1: Enable clock - DMIC1 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_SPARE: Enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.2.88 CLK_RST_CONTROLLER_CLK_ENB_X_SET_0

Offset: 0x284 │ Read/Write: R/W │ Reset: 0x23000780 (0bxx100011000xx00xx00x011110000000)

Bit	Reset	Description
29	0x1	SET_CLK_ENB_PLLG_REF: Clock gate for reference clock branch going to PLLG. 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_PLLA_ADSP: Clock gate for PLLA branch going to ADSP. 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_PLLP_ADSP: Clock gate for PLLP branch going to ADSP. 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_HPLL_ADSP: Clock gate for HPLL(PLLC/PLLC2/PLLC3) branches going to ADSP. 0 = DISABLE 1 = ENABLE
25	0x1	SET_CLK_ENB_DBGAPB: Set enable clock - DBGAPB 0 = DISABLE 1 = ENABLE
24	ENABLE	SET_CLK_ENB_GPU: Set enable clock - GPU 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_SOR1: set enable clock - SOR1 0 = DISABLE 1 = ENABLE
22	DISABLE	SET_CLK_ENB_SOR0: Set enable clock - SOR0 0 = DISABLE 1 = ENABLE
21	DISABLE	SET_CLK_ENB_DPAUX: Set enable clock - DPAUX 0 = DISABLE 1 = ENABLE
18	DISABLE	SET_CLK_ENB_VIC: Set enable clock - VIC 0 = DISABLE 1 = ENABLE
17	DISABLE	SET_CLK_ENB_UART_FST_MIPI_CAL: Set enable clock - UART_FST_MIPI_CAL 0 = DISABLE 1 = ENABLE
14	DISABLE	SET_CLK_ENB_EMC_DLL: Set enable clock - EMC DLL 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_MIPIBIF: Reserved
11	DISABLE	SET_CLK_ENB_VIM2_CLK: Set enable clock - CAMERA ifc 2 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x1	SET_CLK_ENB_MC_BBC: set enable clock - MC BBC 0 = DISABLE 1 = ENABLE
9	0x1	SET_CLK_ENB_MC_CPU: set enable clock - MC CPU 0 = DISABLE 1 = ENABLE
8	0x1	SET_CLK_ENB_MC_CBPA: set enable clock - MC daisy chain2 0 = DISABLE 1 = ENABLE
7	0x1	SET_CLK_ENB_MC_CAPA: set enable clock - MC daisy chain1 0 = DISABLE 1 = ENABLE
6	DISABLE	SET_CLK_ENB_I2C6: Set enable clock - I2C6 0 = DISABLE 1 = ENABLE
5	DISABLE	SET_CLK_ENB_CAM_MCLK2: Set enable clock - CAM_MCLK2 0 = DISABLE 1 = ENABLE
4	DISABLE	SET_CLK_ENB_CAM_MCLK: Set enable clock - CAM_MCLK 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_ETR: set enable ETR clk. 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_DMIC2: set enable clock - DMIC2 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_DMIC1: set enable clock - DMIC1 0 = DISABLE 1 = ENABLE
0	DISABLE	SET_CLK_ENB_SPARE: Set enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.2.89 CLK_RST_CONTROLLER_CLK_ENB_X_CLR_0

Offset: 0x288 │ Read/Write: R/W │ Reset: 0x23000780 (0bxx100011000xx00xx00x011110000000)

Bit	Reset	Description
29	0x1	CLR_CLK_ENB_PLLG_REF: Clock gate for reference clock branch going to PLLG. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_PLLA_ADSP: Clock gate for PLLA branch going to ADSP. 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_PLLP_ADSP: Clock gate for PLLP branch going to ADSP. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_HPLL_ADSP: Clock gate for HPLL(PLLC/PLLC2/PLLC3) branches going to ADSP. 0 = DISABLE 1 = ENABLE
25	0x1	CLR_CLK_ENB_ DBGAPB: Clear enable clock - DBGAPB 0 = DISABLE 1 = ENABLE
24	ENABLE	CLR_CLK_ENB_GPU: Clear enable clock - GPU 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_SOR1: clr enable clock - SOR1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	DISABLE	CLR_CLK_ENB_SOR0: Clear enable clock - SOR0 0 = DISABLE 1 = ENABLE
21	DISABLE	CLR_CLK_ENB_DPAUX: Clear enable clock - DPAUX 0 = DISABLE 1 = ENABLE
18	DISABLE	CLR_CLK_ENB_VIC: Clear enable clock - VIC 0 = DISABLE 1 = ENABLE
17	DISABLE	CLR_CLK_ENB_UART_FST_MIPI_CAL: Clear enable clock - UART_FST_MIPI_CAL 0 = DISABLE 1 = ENABLE
14	DISABLE	CLR_CLK_ENB_EMC_DLL: Clear enable clock - EMC DLL 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_MIPIBIF: Reserved
11	DISABLE	CLR_CLK_ENB_VIM2_CLK: Clear enable clock - CAMERA ifc 2 0 = DISABLE 1 = ENABLE
10	0x1	CLR_CLK_ENB_MC_BBC: clear enable clock - MC BBC 0 = DISABLE 1 = ENABLE
9	0x1	CLR_CLK_ENB_MC_CPU: clear enable clock - MC CPU 0 = DISABLE 1 = ENABLE
8	0x1	CLR_CLK_ENB_MC_CBPA: clear enable clock - MC daisy chain2 0 = DISABLE 1 = ENABLE
7	0x1	CLR_CLK_ENB_MC_CAPA: clear enable clock - MC daisy chain1 0 = DISABLE 1 = ENABLE
6	DISABLE	CLR_CLK_ENB_I2C6: Clear enable clock - I2C6 0 = DISABLE 1 = ENABLE
5	DISABLE	CLR_CLK_ENB_CAM_MCLK2: Clear enable clock - CAM_MCLK2 0 = DISABLE 1 = ENABLE
4	DISABLE	CLR_CLK_ENB_CAM_MCLK: Clear enable clock - CAM_MCLK 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_ETR: clear enable ETR clk. 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_DMIC2: clear enable clock - DMIC2 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_DMIC1: clear enable clock - DMIC1 0 = DISABLE 1 = ENABLE
0	DISABLE	CLR_CLK_ENB_SPARE: Clear enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.2.90 CLK_RST_CONTROLLER_RST_DEVICES_X_0

Offset: 0x28c │ Read/Write: R/W │ Reset: 0x01e42049 (0bxxxxxxx1111xx1xxxx1xxxxxx1xx1xx1)

Bit	Reset	Description
24	ENABLE	SWR_GPU_RST: Reset - GPU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	ENABLE	SWR_SOR1_RST: Reset SOR1 controller. 0 = DISABLE 1 = ENABLE
22	ENABLE	SWR_SOR0_RST: Reset – SOR0 0 = DISABLE 1 = ENABLE
21	ENABLE	SWR_DPAUX_RST: Reset - DPAUX 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_VIC_RST: Reset - VIC 0 = DISABLE 1 = ENABLE
13	ENABLE	SWR_MIPIBIF_RST: Reserved
6	ENABLE	SWR_I2C6_RST: Reset – I2C6 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_ETR_RST: Reset ETR logic. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_SPARE_RST: Reset - SPARE 0 = DISABLE 1 = ENABLE

5.2.91 CLK_RST_CONTROLLER_RST_DEV_X_SET_0

Offset: 0x290 │ Read/Write: R/W │ Reset: 0x01e42049 (0bxxxxxxx1111xx1xxxx1xxxxxx1xx1xx1)

Bit	Reset	Description
24	ENABLE	SET_GPU_RST: Set reset - GPU 0 = DISABLE 1 = ENABLE
23	0x1	SET_SOR1_RST: Set reset - SOR1 0 = DISABLE 1 = ENABLE
22	ENABLE	SET_SOR0_RST: Set reset – SOR0 0 = DISABLE 1 = ENABLE
21	ENABLE	SET_DPAUX_RST: Set reset - DPAUX 0 = DISABLE 1 = ENABLE
18	ENABLE	SET_VIC_RST: Set reset - VIC 0 = DISABLE 1 = ENABLE
13	0x1	SET_MIPIBIF_RST: Reserved
6	ENABLE	SET_I2C6_RST: Set reset – I2C6 0 = DISABLE 1 = ENABLE
3	0x1	SET_ETR_RST: set reset ETR logic 0 = DISABLE 1 = ENABLE
0	0x1	SET_SPARE_RST: Set reset - SPARE 0 = DISABLE 1 = ENABLE

5.2.92 CLK_RST_CONTROLLER_RST_DEV_X_CLR_0

Offset: 0x294 | Read/Write: R/W | Reset: 0x01e42049 (0bxxxxxxx1111xx1xxxx1xxxxxx1xx1xx1)

Bit	Reset	Description
24	ENABLE	CLR_GPU_RST: Clear reset - GPU 0 = DISABLE 1 = ENABLE
23	0x1	CLR_SOR1_RST: Clear reset - SOR1 0 = DISABLE 1 = ENABLE
22	ENABLE	CLR_SOR0_RST: Clear reset – SOR0 0 = DISABLE 1 = ENABLE
21	ENABLE	CLR_DPAUX_RST: Clear reset - DPAUX 0 = DISABLE 1 = ENABLE
18	ENABLE	CLR_VIC_RST: Clear reset - VIC 0 = DISABLE 1 = ENABLE
13	0x1	CLR_MIPIBIF_RST: Reserved
6	ENABLE	CLR_I2C6_RST: Clear reset – I2C6 0 = DISABLE 1 = ENABLE
3	0x1	CLR_ETR_RST: Clear reset ETR logic 0 = DISABLE 1 = ENABLE
0	0x1	CLR_SPARE_RST: Clear reset - SPARE 0 = DISABLE 1 = ENABLE

5.2.93 CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0

Offset: 0x298 | Read/Write: R/W | Reset: 0x00000300 (0b000000xxxxx0000000xxx01100000000)

Bit	Reset	Description
31	DISABLE	CLK_ENB_PLLP_OUT_CPU: Enable PLLP branches to CPU. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_SOR_SAFE: Enable SOR clk. 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_IQC1: Enable IQC1 clk. 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_IQC2: Enable IQC2 clk. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_NVENC: Enable NVENC clk. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_ADSPNEON: Enable ADSP neon clock 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_UARTAPE: Enable UART APE clock. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_QSPI: Enable Quad SPI clk. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_USB2_TRK: Enable USB2 Tracking clock. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17	DISABLE	CLK_ENB_HSIC_TRK: Enable HSIC Tracking clock. 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_VI_I2C: Enable HSIC Tracking clock. 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_DPAUX1: Enable DPAUX1 clk. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_TSECB: Enable TSECB clk. 0 = DISABLE 1 = ENABLE
10	0x0	CLK_ENB_MAUD: enable maud clk. 0 = DISABLE 1 = ENABLE
9	ENABLE	CLK_ENB_MC_CCPA: Enable clock - MC daisy chain3 0 = DISABLE 1 = ENABLE
8	ENABLE	CLK_ENB_MC_CDPA: Enable clock - MC daisy chain4 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_ADSP: Enable ADSP clk. 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_APE: Enable APE clk. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_DMIC3: Enable DMIC3 clk. 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_AXIAP: Enable AXIAP clk. 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_NVJPG: Enable NVJPG clk. 0 = DISABLE 1 = ENABLE
2	DISABLE	CLK_ENB_NVDEC: Enable NVDEC clk. 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_SDMMC_LEGACY_TM: Enable clock - sdmmc_legacy_tm 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_SPARE1: Enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.2.94 CLK_RST_CONTROLLER_CLK_ENB_Y_SET_0

Offset: 0x29c | Read/Write: R/W | Reset: 0x00000300 (0b000000xxxxx0000000xxx01100000000)

Bit	Reset	Description
31	0x0	SET_CLK_ENB_PLLP_OUT_CPU: set enable for PLLP branches going to CPU 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_SOR_SAFE: set enable SOR clk. 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_IQC1: set enable IQC1 clk. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	SET_CLK_ENB_IQC2: set enable IQC2 clk. 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_NVENC: set enable NVENC clk. 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_ADSPNEON: set enable ADSP neon clock 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_UARTAPE: set enable UART APE clock. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CLK_ENB_QSPI: set enable Quad SPI clk. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_USB2_TRK: set enable USB2_TRK clk. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_HSIC_TRK: set enable HSIC_TRK clk. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_VI_I2C: set enable HSIC_TRK clk. 0 = DISABLE 1 = ENABLE
15	DISABLE	SET_CLK_ENB_DPAUX1: Enable DPAUX1 clk. 0 = DISABLE 1 = ENABLE
14	DISABLE	SET_CLK_ENB_TSECB: Enable TSECB clk. 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_MAUD: set enable maud clk. 0 = DISABLE 1 = ENABLE
9	0x1	SET_CLK_ENB_MC_CCPA: set enable clock - MC daisy chain3 0 = DISABLE 1 = ENABLE
8	0x1	SET_CLK_ENB_MC_CDPA: set enable clock - MC daisy chain4 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_ADSP: set enable ADSP clk. 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_APE: set enable APE clk. 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_DMIC3: set enable DMIC3 clk. 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_AXIAP: set enable AXIAP clk. 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_NVJPG: set enable NVJPG clk. 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_NVDEC: set enable NVDEC clk. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_SDMMC_LEGACY_TM: set enable clock - SDMMC_LEGACY_TM 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	SET_CLK_ENB_SPARE1: set enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.2.95 CLK_RST_CONTROLLER_CLK_ENB_Y_CLR_0

Offset: 0x2a0 | Read/Write: R/W | Reset: 0x00000300 (0b000000xxxxx0000000xxx01100000000)

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_PLLP_OUT_CPU: clear enable for PLLP branches going to CPU. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_SOR_SAFE: clear enable SOR clk. 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_IQC1: clear enable IQC1 clk. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_IQC2: clear enable IQC2 clk. 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_NVENC: clear enable NVENC clk. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_ADSPNEON: clear enable ADSP neon clock 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_UARTAPE: clear enable UART APE clk. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_QSPI: clear enable Quad SPI clk. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_USB2_TRK: clear enable USB2_TRK clk. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_HSIC_TRK: clear enable HSIC_TRK clk. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_VI_I2C: clear enable HSIC_TRK clk. 0 = DISABLE 1 = ENABLE
15	DISABLE	CLR_CLK_ENB_DPAUX1: Enable DPAUX1 clk. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLR_CLK_ENB_TSECB: Enable tsecb clk. 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_MAUD: clear enable maud clk. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_CLK_ENB_MC_CCPA: clear enable clock - MC daisy chain3 0 = DISABLE 1 = ENABLE
8	0x1	CLR_CLK_ENB_MC_CDPA: clear enable clock - MC daisy chain4 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_ADSP: clear enable ADSP clk. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	CLR_CLK_ENB_APE: clear enable APE clk. 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_DMIC3: clear enable DMIC3 clk. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_AXIAP: clear enable AXIAP clk. 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_NVJPG: clear enable NVJPG clk. 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_NVDEC: clear enable NVDEC clk. 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_SDMMC_LEGACY_TM: clear enable clock - SDMMC_LEGACY_TM 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_SPARE1: clear enable clock - SPARE 0 = DISABLE 1 = ENABLE

5.2.96 CLK_RST_CONTROLLER_RST_DEVICES_Y_0

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x0fe9f0dd (0bxxxx1111111x1xx11111xxxx11x111x1)

Bit	Reset	Description
27	ENABLE	SWR_NVENC_RST: Reset NVENC logic. 0 = DISABLE 1 = ENABLE
26	ENABLE	SWR_ADSPNEON_RST: Reset ADSP NEON logic. 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_ADSPSCU_RST: Reset ADSP SCU logic. 0 = DISABLE 1 = ENABLE
24	ENABLE	SWR_ADSPWDT_RST: Reset ADSP WDT logic. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_ADSPDBG_RST: Reset ADSP DBG logic. 0 = DISABLE 1 = ENABLE
22	ENABLE	SWR_ADSPPERIPH_RST: Reset ADSP peripheral logic. 0 = DISABLE 1 = ENABLE
21	ENABLE	SWR_ADSPINTF_RST: Reset ADSP Interface logic 0 = DISABLE 1 = ENABLE
19	ENABLE	SWR_QSPI_RST: Reset Quad SPI logic. 0 = DISABLE 1 = ENABLE
16	ENABLE	SWR_VI_I2C_RST: Reset I2C in VI 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_DPAUX1_RST: Reset DPAUX1 logic. 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_TSECB_RST: Reset TSECB logic. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13	ENABLE	SWR_PEX_USB_UPHY_RST: Reset PEX_USB_PAD_PLL logic. 0 = DISABLE 1 = ENABLE
12	ENABLE	SWR_SATA_USB_UPHY_RST: Reset SATA_USB_PAD_PLL logic. 0 = DISABLE 1 = ENABLE
7	ENABLE	SWR_ADSP_RST: Reset ADSP CPU logic. 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_APE_RST: Reset APE logic. 0 = DISABLE 1 = ENABLE
4	ENABLE	SWR_AXIAP_RST: Reset AXIAP logic. 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_NVJPG_RST: Reset NVJPG logic. 0 = DISABLE 1 = ENABLE
2	ENABLE	SWR_NVDEC_RST: Reset NVDEC logic. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_SPARE1_RST: Reset - SPARE 0 = DISABLE 1 = ENABLE

5.2.97 CLK_RST_CONTROLLER_RST_DEV_Y_SET_0

Offset: 0x2a8 | Read/Write: R/W | Reset: 0x0fe9f0dd (0bxxxx1111111x1xx11111xxxx11x111x1)

Bit	Reset	Description
27	0x1	SET_NVENC_RST: set reset NVENC logic 0 = DISABLE 1 = ENABLE
26	0x1	SET_ADSPNEON_RST: set Reset ADSP NEON logic. 0 = DISABLE 1 = ENABLE
25	0x1	SET_ADSPSCU_RST: set Reset ADSP SCU logic. 0 = DISABLE 1 = ENABLE
24	0x1	SET_ADSPWDT_RST: set Reset ADSP WDT logic. 0 = DISABLE 1 = ENABLE
23	0x1	SET_ADSPDBG_RST: set Reset ADSP DBG logic. 0 = DISABLE 1 = ENABLE
22	0x1	SET_ADSPPERIPH_RST: set Reset ADSP periph logic. 0 = DISABLE 1 = ENABLE
21	0x1	SET_ADSPINTF_RST: set Reset ADSP Interface logic 0 = DISABLE 1 = ENABLE
19	0x1	SET_QSPI_RST: set Reset Quad SPI logic. 0 = DISABLE 1 = ENABLE
16	0x1	SET_VI_I2C_RST: set Reset I2C logic in VI. 0 = DISABLE 1 = ENABLE
15	ENABLE	SET_DPAUX1_RST: Reset DPAUX1 logic. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	ENABLE	SET_TSECB_RST: Reset TSECB logic. 0 = DISABLE 1 = ENABLE
13	0x1	SET_PEX_USB_UPHY_RST: set PEX_USB_PAD_PLL RESET logic 0 = DISABLE 1 = ENABLE
12	0x1	SET_SATA_USB_UPHY_RST: set SATA_USB_PAD_PLL RESET logic 0 = DISABLE 1 = ENABLE
7	0x1	SET_ADSP_RST: set Reset ADSP CPU logic. 0 = DISABLE 1 = ENABLE
6	0x1	SET_APE_RST: set reset APE logic 0 = DISABLE 1 = ENABLE
4	0x1	SET_AXIAP_RST: set reset AXIAP logic 0 = DISABLE 1 = ENABLE
3	0x1	SET_NVJPG_RST: set reset NVJPG logic 0 = DISABLE 1 = ENABLE
2	0x1	SET_NVDEC_RST: set reset NVDEC logic 0 = DISABLE 1 = ENABLE
0	0x1	SET_SPARE1_RST: set reset - SPARE 0 = DISABLE 1 = ENABLE

5.2.98 CLK_RST_CONTROLLER_RST_DEV_Y_CLR_0

Offset: 0x2ac | Read/Write: R/W | Reset: 0x0fe9f0dd (0bxxxx1111111x1xx11111xxxx11x111x1)

Bit	Reset	Description
27	0x1	CLR_NVENC_RST: clear reset NVENC logic 0 = DISABLE 1 = ENABLE
26	0x1	CLR_ADSPNEON_RST: clear reset ADSP NEON logic. 0 = DISABLE 1 = ENABLE
25	0x1	CLR_ADSPSCU_RST: clear reset ADSP SCU logic. 0 = DISABLE 1 = ENABLE
24	0x1	CLR_ADSPWDT_RST: clear reset ADSP WDT logic. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_ADSPDBG_RST: clear reset ADSP DBG logic. 0 = DISABLE 1 = ENABLE
22	0x1	CLR_ADSPPERIPH_RST: clear reset ADSP periph logic. 0 = DISABLE 1 = ENABLE
21	0x1	CLR_ADSPINTF_RST: clear reset ADSP Interface logic 0 = DISABLE 1 = ENABLE
19	0x1	CLR_QSPI_RST: clear reset Quad SPI logic 0 = DISABLE 1 = ENABLE
16	0x1	CLR_VI_I2C_RST: clear reset I2C logic in VI 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	ENABLE	CLR_DPAUX1_RST: Reset DPAUX1 logic. 0 = DISABLE 1 = ENABLE
14	ENABLE	CLR_TSECB_RST: Reset TSECB logic. 0 = DISABLE 1 = ENABLE
13	0x1	CLR_PEX_USB_UPHY_RST: Clear PEX_USB_PAD_PLL RESET logic 0 = DISABLE 1 = ENABLE
12	0x1	CLR_SATA_USB_UPHY_RST: Clear SATA_USB_PAD_PLL RESET logic 0 = DISABLE 1 = ENABLE
7	0x1	CLR_ADSP_RST: clear reset ADSP CPU logic. 0 = DISABLE 1 = ENABLE
6	0x1	CLR_APE_RST: clear reset APE logic 0 = DISABLE 1 = ENABLE
4	0x1	CLR_AXIAP_RST: clear reset AXIAP logic 0 = DISABLE 1 = ENABLE
3	0x1	CLR_NVJPG_RST: clear reset NVJPG logic 0 = DISABLE 1 = ENABLE
2	0x1	CLR_NVDEC_RST: clear reset NVDEC logic 0 = DISABLE 1 = ENABLE
0	0x1	CLR_SPARE1_RST: clear reset - SPARE 0 = DISABLE 1 = ENABLE

5.2.99 CLK_RST_CONTROLLER_DFLL_BASE_0

Offset: 0x2f4 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	ENABLE	DVFS_DFLL_RESET: Assert the reset to DFLL. 0 = DISABLE 1 = ENABLE

5.2.100 CLK_RST_CONTROLLER_RST_DEV_L_SET_0

The RST_DEV_(L,H,U, V, W, X, Y)_(SET,CLR) and CLK_ENB_(L,H,U, V, W, X, Y)_(SET,CLR) registers are provided as an alternate method of programming the same registers found in RST_DEVICES_(L,H,U, V, W, X, Y) and CLK_OUT_ENB_(L,H,U, V, W, X, Y) registers, respectively.

Therefore, using either methods will change the same underlying peripherals reset, and clock enable control for writes. For reads, you can use either method to retrieve the reset/clock-enable state of each peripheral.

Offset: 0x300 │ Read/Write: R/W │ Reset: 0x1cd3dXXX (0b0xx111xx11x1xx1111x1xx1x11xx100x)

Bit	R/W	Reset	Description
31	RW	0x0	SET_CACHE2_RST: Set reset for the BPMP-Lite cache controller. 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_HOST1X_RST: Set reset for the HOST1X. 0 = DISABLE 1 = ENABLE
27	RW	0x1	SET_DISP1_RST: Set reset for the DISP1 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
26	RW	0x1	SET_DISP2_RST: Set reset for the DISP2 controller. 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_ISP_RST: Set reset for the ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	SET_USBD_RST: Set reset for the USB controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	SET_VI_RST: Set reset for the VI controller. 0 = DISABLE 1 = ENABLE
17	RW	0x1	SET_PWM_RST: Set reset for the Pulse Width Modulator. 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_SDMMC4_RST: Set reset for the SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_SDMMC1_RST: Set reset for the SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_I2C1_RST: Set reset for the I2C1 controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SDMMC2_RST: Set reset for the SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	RW	0x0	SET_GPIO_RST: Set reset for the GPIO controller. 0 = DISABLE 1 = ENABLE
7	RW	0x1	SET_UARTB_RST: Set reset for the UARTB/VFIR controller. 0 = DISABLE 1 = ENABLE
6	RW	0x1	SET_UARTA_RST: Set reset for the UARTA controller. 0 = DISABLE 1 = ENABLE
5	RO	x	SET_TMR_RST: Reserved
3	RW	0x1	SET_ISPB_RST: Set reset - ISPB
2	RW	0x0	SET_TRIG_SYS_RST: Write 1 to pulse the System Reset signal. 0 = DISABLE 1 = ENABLE
1	RW	0x0	SET_COP_RST: Set/reset BPMP-Lite. 0 = DISABLE 1 = ENABLE
0	RO	x	SET_CPU_RST: Deprecated. This bit can be written but it always reads as 0.

5.2.101 CLK_RST_CONTROLLER_RST_DEV_L_CLR_0

Offset: 0x304 | Read/Write: R/W | Reset: 0x1cd3dXXX (0b0xx111xx11x1xx1111x1xx1x11xx1x0x)

Bit	R/W	Reset	Description
31	RW	0x0	CLR_CACHE2_RST: Clear reset for the BPMP-Lite cache controller. 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_HOST1X_RST: Clear reset for the HOST1X. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
27	RW	0x1	CLR_DISP1_RST: Clear reset for the DISP1 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	CLR_DISP2_RST: Clear reset for the DISP2 controller. 0 = DISABLE 1 = ENABLE.
23	RW	0x1	CLR_ISP_RST: Clear reset for the ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	CLR_USBD_RST: Clear reset for the USB controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	CLR_VI_RST: Clear reset for the VI controller. 0 = DISABLE 1 = ENABLE
17	RW	0x1	CLR_PWM_RST: Clear reset for the Pulse Width Modulator. 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_SDMMC4_RST: Clear reset for the SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_SDMMC1_RST: Clear reset for the SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	CLR_I2C1_RST: Clear reset for the I2C1 controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	CLR_SDMMC2_RST: Clear reset for the SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	RO	X	CLR_GPIO_RST: Clear reset for the GPIO controller. 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_UARTB_RST: Clear reset for the UARTB/VFIR controller. 0 = DISABLE 1 = ENABLE
6	RW	0x1	CLR_UARTA_RST: Clear reset for the UARTA controller. 0 = DISABLE 1 = ENABLE
5	RO	X	CLR_TMR_RST: Reserved
3	RW	0x1	CLR_ISPB_RST: Clear reset - ISBP 0 = DISABLE 1 = ENABLE
1	RW	0x0	CLR_COP_RST: Clear reset for the BPMP-Lite. 0 = DISABLE 1 = ENABLE
0	RW	X	CLR_CPU_RST: Deprecated. This bit can be written but it always reads as 0.

5.2.102 CLK_RST_CONTROLLER_RST_DEV_H_SET_0

Offset: 0x308 | Read/Write: R/W | Reset: 0x87d1f32X (0b1xxxx11111x1xxx11111xx110x1xx11x)

Bit	R/W	Reset	Description
31	RW	0x1	SET_BSEV_RST: Set reset for the BSEV controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
26	RW	0x1	SET_USB2_RST: Set reset for the USB2 controller. 0 = DISABLE 1 = ENABLE
25	RW	0x1	SET_EMC_RST: Set reset for the EMC controller. 0 = DISABLE 1 = ENABLE
24	RW	0x1	SET_MIPI_CAL_RST: Set reset for the MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_UARTC_RST: Set reset for the UARTC controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	SET_I2C2_RST: Set reset for the I2C2 controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	SET_CSI_RST: Set reset for the CSI controller. 0 = DISABLE 1 = ENABLE
16	RW	0x1	SET_DSI_RST: Set reset for the DSI controller. 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_ I2C5_RST: Set reset for the I2C5 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_SPI3_RST: Set reset for the SPI3 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_SPI2_RST: Set reset for the SPI2 controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SPI1_RST: Set reset for the SPI1 Controller. 0 = DISABLE 1 = ENABLE
8	RW	0x1	SET_KFUSE_RST: Set reset for the KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	0x1	SET_STAT_MON_RST: Set reset for the statistic monitor 0 = DISABLE 1 = ENABLE
2	RW	0x1	SET_APBDMA_RST: Set reset for the APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	0x1	SET_AHBDMA_RST: Set reset for the AHB-DMA. 0 = DISABLE 1 = ENABLE
0	RO	X	SET_MEM_RST: Set reset MC. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.2.103 CLK_RST_CONTROLLER_RST_DEV_H_CLR_0

Offset: 0x30c │ Read/Write: R/W │ Reset: 0x87d1f327 (0b1xxxx11111x1xxx11111xx110x1xx111)

Bit	Reset	Description
31	0x1	CLR_BSEV_RST: Clear reset for the BSEV controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	0x1	CLR_USB2_RST: Clear reset for the USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x1	CLR_EMC_RST: Clear reset for the EMC controller. 0 = DISABLE 1 = ENABLE
24	0X1	CLR_MIPI_CAL_RST: Clear reset for the MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_UARTC_RST: Clear reset for the UARTC controller. 0 = DISABLE 1 = ENABLE
22	0x1	CLR_I2C2_RST: Clear reset for the I2C2 controller. 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CSI_RST: Clear reset for the CSI controller. 0 = DISABLE 1 = ENABLE
16	0x1	CLR_DSI_RST: Clear reset for the DSI controller. 0 = DISABLE 1 = ENABLE
15	0x1	CLR_ I2C5_RST: Clear reset for the I2C5 controller. 0 = DISABLE 1 = ENABLE
14	0x1	CLR_SPI3_RST: Clear reset for the SPI3 controller. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_SPI2_RST: Clear reset for the SPI2 controller. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_SPI1_RST: Clear reset for the SPI1 controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_KFUSE_RST: Clear reset for the KFuse controller. 0 = DISABLE 1 = ENABLE
5	0x1	CLR_STAT_MON_RST: Clear reset for the statistic monitor. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_APBDMA_RST: Clear reset for the APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_AHBDMA_RST: Clear reset for the AHB-DMA. 0 = DISABLE 1 = ENABLE
0	X	CLR_MEM_RST: Clear reset for the MC. Note that MC reset can only be cleared by software for security reasons. 0 = DISABLE 1 = ENABLE

5.2.104 CLK_RST_CONTROLLER_RST_DEV_U_SET_0

Offset: 0x310 | Read/Write: R/W | Reset: 0x828ec5fa (0b1xxxxx1x1xxx111x11x0x10111111x1x)

Bit	Reset	Description
31	0x1	SET_XUSB_DEV_RST: Set reset for the XUSB DEV logic. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x1	SET_XUSB_HOST_RST: Set reset for the XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	0x1	SET_EMUCIF_RST: Set reset for the EMUCIF logic 0 = DISABLE 1 = ENABLE
19	0x1	SET_TSEC_RST: Set reset for the TSEC logic 0 = DISABLE 1 = ENABLE
18	0x1	SET_DSIB_RST: Set reset for the DSIB controller 0 = DISABLE 1 = ENABLE
17	0x1	SET_I2C_SLOW_RST: Set reset for the I2C_SLOW 0 = DISABLE 1 = ENABLE
15	0x1	SET_DTV_RST: Set reset for the DTV controller. 0 = DISABLE 1 = ENABLE
14	0x1	SET_SOC_THERM_RST: Set reset for the SOC_THERM controller. 0 = DISABLE 1 = ENABLE
10	0x1	SET_PCIEXCLK_RST: Set reset for the PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	0x0	SET_CSITE_RST: Set reset for the CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	SET_AFI_RST: Set reset for the AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	Unused, reserved.
6	0x1	SET_PCIE_RST: Set reset for the PCIe controller. 0 = DISABLE 1 = ENABLE
5	0x1	SET_SDMMC3_RST: Set reset for the SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	SET_SPI4_RST: Set reset for the SPI4 controller. 0 = DISABLE 1 = ENABLE
3	0x1	SET_I2C3_RST: Set reset for the I2C3 controller. 0 = DISABLE 1 = ENABLE
1	0x1	SET_UARTD_RST: Set reset for the UARTD controller. 0 = DISABLE 1 = ENABLE

5.2.105 CLK_RST_CONTROLLER_RST_DEV_U_CLR_0

Offset: 0x314 | Read/Write: R/W | Reset: 0x828ec5fa (0b1xxxxx1x1xxx111x11x0x10111111x1x)

Bit	Reset	Description
31	0x1	CLR_XUSB_DEV_RST: Clear reset for the XUSB DEV logic. 0 = DISABLE 1 = ENABLE
25	0x1	CLR_XUSB_HOST_RST: Clear reset for the XUSB HOST logic. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x1	CLR_EMUCIF_RST: Clear reset for the EMUCIF logic 0 = DISABLE 1 = ENABLE
19	0x1	CLR_TSEC_RST: Clear reset for the TSEC logic 0 = DISABLE 1 = ENABLE
18	0x1	CLR_DSIB_RST: Clear reset for the DSIB Controller 0 = DISABLE 1 = ENABLE
17	0x1	CLR_I2C_SLOW_RST: Clear reset for the I2C_SLOW 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DTV_RST: Clear reset for the DTV controller 0 = DISABLE 1 = ENABLE
14	0x1	CLR_SOC_THERM_RST: Clear reset for the SOC_THERM controller. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_PCIEXCLK_RST: Clear reset for the PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CSITE_RST: Clear reset for the CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_AFI_RST: Clear reset for the AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	Unused, reserved.
6	0x1	CLR_PCIE_RST: Clear reset for the PCIe controller. 0 = DISABLE 1 = ENABLE
5	0x1	CLR_SDMMC3_RST: Clear reset for the SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	CLR_SPI4_RST: Clear reset for the SPI4 controller. 0 = DISABLE 1 = ENABLE
3	0x1	CLR_I2C3_RST: Clear reset for the I2C3 controller. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_UARTD_RST: Clear reset for the UARTD controller. 0 = DISABLE 1 = ENABLE

5.2.106 CLK_RST_CONTROLLER_CLK_ENB_L_SET_0

Offset: 0x320 | Read/Write: R/W | Reset: 0x80000130 (0b10x000xx00x0x00000x0000100110xx0)

Bit	Reset	Description
31	0x1	SET_CLK_ENB_CACHE2: Set enable clock to BPMP-Lite cache controller. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_I2S1: Set enable clock to I2S1 controller 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_HOST1X: Set enable clock to HOST1X. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	SET_CLK_ENB_DISP1: Set enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_DISP2: Set enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_ISP: Set enable clock to ISP controller. 0 = DISABLE 1 = ENABLE
22	0x0	SET_CLK_ENB_USBD: Set enable clock to USB controller 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_VI: Set enable clock to VI controller. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_I2S3: Set enable clock to I2S3 controller 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_PWM: Set enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_SDMMC4: Set enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SDMMC1: Set enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_I2C1: Set enable clock to I2C1 controller 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_I2S2: Set enable clock to I2S2 controller 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_SPDIF: Set enable clock to S/PDIF controller 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SDMMC2: Set enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	0x1	SET_CLK_ENB_GPIO: Set enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_UARTB: Set enable clock to UARTB/VFIR controller 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_UARTA: Set enable clock to UARTA controller 0 = DISABLE 1 = ENABLE
5	0x1	SET_CLK_ENB_TMR: Set enable clock to Timer controller 0 = DISABLE 1 = ENABLE
4	0x1	SET_CLK_ENB_RTC: Set enable clock to RTC controller 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_ISPB: Set enable clock - ISPB 0 = DISABLE 1 = ENABLE

5.2.107 CLK_RST_CONTROLLER_CLK_ENB_L_CLR_0

Offset: 0x324 | Read/Write: R/W | Reset: 0x80000130 (0b10x000xx00x0x00000x0000100110xx0)

Bit	Reset	Description
31	0x1	CLR_CLK_ENB_CACHE2: Clear enable clock to BPMP-Lite cache controller. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_I2S1: Clear enable clock to I2S1 controller 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_HOST1X: Clear enable clock to HOST1X. 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_DISP1: Clear enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_DISP2: Clear enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_ISP: Clear enable clock to ISP controller. 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CLK_ENB_USBD: Clear enable clock to USB controller 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_VI: Clear enable clock to VI controller. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_I2S3: Clear enable clock to I2S3 controller 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_PWM: Clear enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_SDMMC4: Clear enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SDMMC1: Clear enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_I2C1: Clear enable clock to I2C1 controller 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_I2S2: Clear enable clock to I2S2 controller 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_SPDIF: Clear enable clock to S/PDIF controller 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_SDMMC2: Clear enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_CLK_ENB_GPIO: Clear enable clock to GPIO controller 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_UARTB: Clear enable clock to UARTB/VFIR controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	CLR_CLK_ENB_UARTA: Clear enable clock to UARTA controller 0 = DISABLE 1 = ENABLE
5	0x1	CLR_CLK_ENB_TMR: Clear enable clock to Timer controller 0 = DISABLE 1 = ENABLE
4	0x1	CLR_CLK_ENB_RTC: Clear enable clock to RTC controller 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_ISPB: Clear enable clock - ISPB 0 = DISABLE 1 = ENABLE

5.2.108 CLK_RST_CONTROLLER_CLK_ENB_H_SET_0

Offset: 0x328 | Read/Write: R/W | Reset: 0x00000080 (0b0xxxx00000x0xxx00000xx00100xx000)

Bit	Reset	Description
31	0x0	SET_CLK_ENB_BSEV: Set enable clock to BSEV controller. 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_USB2: Set enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_EMC: Set enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
24	0x0	SET_CLK_ENB_MIPI_CAL: Set enable clock to MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_UARTC: Set enable clock to UARTC controller 0 = DISABLE 1 = ENABLE
22	0x0	SET_CLK_ENB_I2C2: Set enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_CSI: Set enable clock to CSI controller. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_DSI: Set enable clock to DSI controller. 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_ I2C5: Set enable clock to I2C5 controller. 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SPI3: Set enable clock to SPI 3 controller. 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_SPI2: Set enable clock to SPI 2 controller. 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SPI1: Set enable clock to SPI1 controller. 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_KFUSE: Set enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x1	SET_CLK_ENB_FUSE: Set enable clock to FUSE controller and host2jtag clock (also used for RAM repair). 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_PMC: Set enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_STAT_MON: Set enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_APBDMA: Set enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_AHBDMA: Set enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_MEM: Set enable clock to MC. 0 = DISABLE 1 = ENABLE

5.2.109 CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0

Offset: 0x32c │ Read/Write: R/W │ Reset: 0x00000080 (0b0xxxx00000x0xxx00000xx00100xx000)

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_BSEV: Clear enable clock to BSEV controller. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_USB2: Clear enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_EMC: Clear enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
24	0x0	CLR_CLK_ENB_MIPI_CAL: Clear enable clock to MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_UARTC: Clear enable clock to UARTC controller. 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CLK_ENB_I2C2: Clear enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_CSI: Clear enable clock to CSI controller. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_DSI: Clear enable clock to DSI controller. 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_ I2C5: Clear enable clock to -I2C5 controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SPI3: Clear enable clock to SPI3 controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SPI2: Clear enable clock to SPI2 Controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	CLR_CLK_ENB_SPI1: Clear enable clock to SPI1 Controller. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_KFUSE: Clear enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	0x1	CLR_CLK_ENB_FUSE: Clear enable clock to FUSE controller and host2jtag clock (also used for RAM repair). 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_PMC: Clear enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_STAT_MON: Clear enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_APBDMA: Clear enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_AHBDMA: Clear enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_MEM: Clear enable clock to MC 0 = DISABLE 1 = ENABLE

5.2.110 CLK_RST_CONTROLLER_CLK_ENB_U_SET_0

Offset: 0x330 | Read/Write: R/W | Reset: 0x01f80200 (0b0000xx011111100x00x0xx1000000x0x)

Bit	Reset	Description
31	0x0	SET_CLK_ENB_XUSB_DEV: Set enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_DEV1_OUT: Set enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_DEV2_OUT: Set enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_SUS_OUT: Set enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_XUSB_HOST: Set enable clock to XUSB HOST. 0 = DISABLE 1 = ENABLE
24	0x1	SET_CLK_ENB_CRAM2: Set enable clock for the BPMP-Lite cache RAM clock. 0 = DISABLE 1 = ENABLE
23	0x1	SET_CLK_ENB_IRAMD: Set enable clock for the IRAMD clock. 0 = DISABLE 1 = ENABLE
22	0x1	SET_CLK_ENB_IRAMC: Set enable clock for the IRAMC clock. 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_IRAMB: Set enable clock for the IRAMB clock. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x1	SET_CLK_ENB_IRAMA: Set enable clock for the IRAMB clock. 0 = DISABLE 1 = ENABLE
19	0x1	SET_CLK_ENB_TSEC: Set enable clock for the TSEC clock. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_DSIB: Set enable clock to DSIB controller 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_I2C_SLOW: Set enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_DTV: Set enable clock to DTV Controller. Moved to U:15 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SOC_THERM: Set enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_LA: set enable clock to LA. 0 = DISABLE 1 = ENABLE
9	0x1	SET_CLK_ENB_CSITE: Set enable clock to CSITE. 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_AFI: Set enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	0x0	Unused, reserved.
6	0x0	SET_CLK_ENB_PCIE: Set enable clock to PCIe. 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_SDMMC3: Set enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_SPI4: Set enable clock to SPI4. 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_I2C3: Set enable clock to I2C3. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_UARTD: Set enable clock to UARTD. 0 = DISABLE 1 = ENABLE

5.2.111 CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0

Offset: 0x334 | Read/Write: R/W | Reset: 0x01f00200 (0b0000xx011111000x00x0xx1000000x0x)

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_XUSB_DEV: Clear enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_DEV1_OUT: Clear enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_DEV2_OUT: Clear enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	CLR_CLK_ENB_SUS_OUT: Clear enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_XUSB_HOST: Clear enable clock to XUSB HOST. 0 = DISABLE 1 = ENABLE
24	0x1	CLR_CLK_ENB_CRAM2: Clear enable clock to the BPMP-Lite cache RAM clock. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_CLK_ENB_IRAMD: Clear enable clock to the IRAMD clock. 0 = DISABLE 1 = ENABLE
22	0x1	CLR_CLK_ENB_IRAMC: Clear enable clock to the IRAMC clock. 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_IRAMB: Clear enable clock to the IRAMB clock. 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CLK_ENB_IRAMA: Clear enable clock to the IRAMB clock. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_TSEC: Clear enable clock to the TSEC clock. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_DSIB: Clear enable clock to DSIB controller. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_I2C_SLOW: Clear enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_DTV: Clear enable clock to DTV controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SOC_THERM: Clear enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLK_ENB_LA: clear enable clock to LA. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_CLK_ENB_CSITE: Clear enable clock to CSITE. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_AFI: Clear enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	0x0	Unused, reserved.
6	0x0	CLR_CLK_ENB_PCIE: Clear enable clock to PCIe. 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_SDMMC3: Clear enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_SPI4: Clear enable clock to SPI4. 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_I2C3: Clear enable clock to I2C3. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	CLR_CLK_ENB_UARTD: Clear enable clock to UARTD. 0 = DISABLE 1 = ENABLE

5.2.112 CLK_RST_CONTROLLER_CCPLEX_PG_SM_OVRD_0

CCPLEX Power Gate State-Machine Overrides

This register provides the override controls for the state-machine outputs.

Offset: 0x33c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxx000000xx000000xx000000)

Bit	Reset	Description
21	0x0	CLKSTOP_OVRD_FCPU_RAIL: 1 = Clamp the corresponding clocks for the active cluster at the time 0 = DISABLE 1 = ENABLE
20	0x0	CLKSTOP_OVRD_FCPU_NC: 1 = Clamp the corresponding clocks for the active cluster at the time 0 = DISABLE 1 = ENABLE
19	0x0	CLKSTOP_OVRD_FCPU_3: 1 = Clamp the corresponding clocks for the active cluster at the time 0 = DISABLE 1 = ENABLE
18	0x0	CLKSTOP_OVRD_FCPU_2: 1 = Clamp the corresponding clocks for the active cluster at the time 0 = DISABLE 1 = ENABLE
17	0x0	CLKSTOP_OVRD_FCPU_1: 1 = Clamp the corresponding clocks for the active cluster at the time 0 = DISABLE 1 = ENABLE
16	0x0	CLKSTOP_OVRD_FCPU_0: 1 = Clamp the corresponding clocks for the active cluster at the time 0 = DISABLE 1 = ENABLE
13	0x0	RST_OVRD_FCPU_RAIL: 1 = Assert the corresponding reset for the active cluster at the time 0 = DISABLE 1 = ENABLE
12	0x0	RST_OVRD_FCPU_NC: 1 = Assert the corresponding reset for the active cluster at the time 0 = DISABLE 1 = ENABLE
11	0x0	RST_OVRD_FCPU_3: 1 = Assert the corresponding reset for the active cluster at the time 0 = DISABLE 1 = ENABLE
10	0x0	RST_OVRD_FCPU_2: 1 = Assert the corresponding reset for the active cluster at the time 0 = DISABLE 1 = ENABLE
9	0x0	RST_OVRD_FCPU_1: 1 = Assert the corresponding reset for the active cluster at the time 0 = DISABLE 1 = ENABLE
8	0x0	RST_OVRD_FCPU_0: 1 = Assert the corresponding reset for the active cluster at the time 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	EN_RST_CG_OVRD_FCPU_RAIL: 1 = Select the OVRD controls in this register for the active cluster at the time 0 = DISABLE 1 = ENABLE
4	0x0	EN_RST_CG_OVRD_FCPU_NC: 1 = Select the OVRD controls in this register for the active cluster at the time 0 = DISABLE 1 = ENABLE
3	0x0	EN_RST_CG_OVRD_FCPU_3: 1 = Select the OVRD controls in this register for the active cluster at the time 0 = DISABLE 1 = ENABLE
2	0x0	EN_RST_CG_OVRD_FCPU_2: 1 = Select the OVRD controls in this register for the active cluster at the time 0 = DISABLE 1 = ENABLE
1	0x0	EN_RST_CG_OVRD_FCPU_1: 1 = Select the OVRD controls in this register for the active cluster at the time 0 = DISABLE 1 = ENABLE
0	0x0	EN_RST_CG_OVRD_FCPU_0: 1 = Select the OVRD controls in this register for the active cluster at the time 0 = DISABLE 1 = ENABLE

5.2.113 CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET_0

Note: When transitioning to a new 4-bit value, all DBGRESET bits in this register which are transitioning to 1 must be set before all DBGRESET bits which are transitioning to 0 are cleared.

Offset: 0x340 | Read/Write: R/W | Reset: 0x2000feef (0bx010xxx00000000011111111011101111)

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = Assert nPRESETDBG to the debug APB interface. 0 = DISABLE 1 = ENABLE
29	0x1	SET_SCURESET: 1 = Assert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: Reserved.
24	0x0	SET_L2RESET: 1 = Assert nL2RESET to the CPU. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CXRESET3: Reserved
22	0x0	SET_CXRESET2: Reserved
21	0x0	SET_CXRESET1: Reserved
20	0x0	SET_CXRESET0: Reserved
19	0x0	SET_CORERESET3: 1 = Assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CORERESET2: 1 = Assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CORERESET1: 1 = Assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CORERESET0: 1 = Assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	SET_DBGRESET3: Reserved
14	0x1	SET_DBGRESET2: Reserved

Bit	Reset	Description
13	0x1	SET_DBGRESET1: Reserved
12	0x1	SET_DBGRESET0: Reserved
11	0x1	SET_WDRESET3: Reserved.
10	0x1	SET_WDRESET2: Reserved.
9	0x1	SET_WDRESET1: Reserved.
8	0x0	SET_WDRESET0: Reserved.
7	0x1	SET_DERESET3: Reserved.
6	0x1	SET_DERESET2: Reserved.
5	0x1	SET_DERESET1: Reserved.
4	0x0	SET_DERESET0: Reserved.
3	0x1	SET_CPURESET3: 1 = assert nCPUPORESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = assert nCPUPORESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPURESET1: 1 = assert nCPUPORESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	SET_CPURESET0: 1 = assert nCPUPORESET to CPU0. 0 = DISABLE 1 = ENABLE

5.2.114 CLK_RST_CONTROLLER_RST_DEVICES_V_0

Expanded Register Set

Bank V for required clock reset register changes. A gap is left after bank V to allow another bank, if necessary.

Offset: 0x358 │ Read/Write: R/W │ Reset: 0xff81808X (0b111111111xxxxxx11xxxxxxx1xxx0xxx)

Bit	R/W	Reset	Description
29	RW	ENABLE	SWR_HDA_RST: Reset High Definition Audio 0 = DISABLE 1 = ENABLE
28	RW	ENABLE	SWR_SATA_RST: Reset SATA 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_ACTMON_RST: Reset ACTMON 0 = DISABLE 1 = ENABLE
16	RW	ENABLE	SWR_ATOMICS_RST: Reset ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_HDA2CODEC_2X_RST: Reset HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SWR_I2C4_RST: Reset I2C4 0 = DISABLE 1 = ENABLE
3	RW	DISABLE	SWR_MSELECT_RST: Reset MSELECT 0 = DISABLE 1 = ENABLE
1	RO	X	SWR_CPULP_RST: Reserved.
0	RO	X	SWR_CPUG_RST: Reserved.

5.2.115 CLK_RST_CONTROLLER_RST_DEVICES_W_0

Offset: 0x35c | Read/Write: R/W | Reset: 0x190077ff (0bxxx11xx1xx0xxxxxx111x11111111111)

Bit	Reset	Description
28	ENABLE	SWR_XUSB_SS_RST: Reset XUSB_SS logic. 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_DVFS_RST: Reset CLDVFS 0 = DISABLE 1 = ENABLE
21	DISABLE	SWR_ENTROPY_RST: Reset Entropy logic. 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_XUSB_PADCTL_RST: Reset XUSB PADCTL logic. IOBIST control has clock enable only. 0 = DISABLE 1 = ENABLE
13	0x1	SWR_RESERVED10_RST: Reserved
12	0x1	SWR_RESERVED9_RST: Reserved
10	0x1	SWR_RESERVED7_RST: Reserved
9	0x1	SWR_RESERVED6_RST: Reserved
8	ENABLE	SWR_CEC_RST: Reset CEC 0 = DISABLE 1 = ENABLE
7	0x1	SWR_RESERVED5_RST: Reserved
6	0x1	SWR_RESERVED4_RST: Reserved
5	0x1	SWR_RESERVED3_RST: Reserved
4	0x1	SWR_RESERVED2_RST: Reserved
3	0x1	SWR_RESERVED1_RST: Reserved
2	0x1	SWR_RESERVED0_RST: Reserved
1	ENABLE	SWR_SATACOLD_RST: Reset SATACOLD 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_HDA2HDMICODEC_RST: Reset HDA2HDMICODEC (no dedicated module or clock.). 0 = DISABLE 1 = ENABLE

5.2.116 CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0

Offset: 0x360 | Read/Write: R/W | Reset: 0x00400008 (0b0000000001xxxxx00xxx00xx00001x00)

Bit	Reset	Description
29	DISABLE	CLK_ENB_HDA: Enable clock to High Definition Audio 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_SATA: Enable clock to SATA 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_SATA_OOB: Enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_EXTPERIPH3: Enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_EXTPERIPH2: Enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	DISABLE	CLK_ENB_EXTPERIPH1: Enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	DISABLE	CLK_ENB_ACTMON: Enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	ENABLE	CLK_ENB_SPDIF_DOUBLER: Enable S/PDIF audio sync clock doubler. 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_ATOMICS: Enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_HDA2CODEC_2X: Enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_APB2APE: Enable APB clock to APE 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_AHUB: Enable clock to AHUB 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_I2C4: Enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_I2S5: Enable clock to I2S5 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_I2S4: Enable clock to I2S4 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_TSENSOR: Enable clock to TSENSOR 0 = DISABLE 1 = ENABLE
3	ENABLE	CLK_ENB_MSELECT: Enable clock to MSELECT 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_CPUG: Enable clock to CPUG. 0 = DISABLE 1 = ENABLE

5.2.117 CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x402000fc (0bx1000xx0xx1000000x00x00011111100)

Bit	Reset	Description
30	ENABLE	CLK_ENB_MC1: Enable clock to MC1 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_EMC_LATENCY: Enable clock to EMC latency 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_XUSB_SS: Enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_DVFS: Enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_ENTROPY: Enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_DSIB_LP: Enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	DISABLE	CLK_ENB_DSIA_LP: Enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_CILEF: Enable clock to CSI CILEF. 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_CILCD: Enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_CILAB: Enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_XUSB: Enable clock to XUSB. IOBIST control has clock enable only. 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_MIPI_IOBIST: Enable clock to MIPI IOBIST. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_SATA_IOBIST: Enable clock to SATA IOBIST. 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_EMC_IOBIST: Enable clock to EMC IOBIST. 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_PCIE2_IOBIST: Enable clock to PCIE2 IOBIST. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_CEC: Enable clock to CEC. 0 = DISABLE 1 = ENABLE
7	0x1	CLK_ENB_PCIERX5: Enable clock to PCIERX5. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
6	0x1	CLK_ENB_PCIERX4: Enable clock to PCIERX4. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
5	0x1	CLK_ENB_PCIERX3: Enable clock to PCIERX3. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
4	0x1	CLK_ENB_PCIERX2: Enable clock to PCIERX2. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
3	0x1	CLK_ENB_PCIERX1: Enable clock to PCIERX1. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
2	0x1	CLK_ENB_PCIERX0: Enable clock to PCIERX0. CLK_ENB_PCIE is also the master control for all PCIERX* clocks.
1	DISABLE	CLK_ENB_RESERVED0: Unused clock enable for satacoldrstn 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_HDA2HDMICODEC: Enable clock to HDA2HDMICODEC (no dedicated module or clock). 0 = DISABLE 1 = ENABLE

5.2.118 CLK_RST_CONTROLLER_CCLKG_BURST_POLICY_0

Offset: 0x368 | Read/Write: R/W | Reset: 0x10XX0000 (0b00010000xxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:28	RW	0x1	CPU_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ

Bit	R/W	Reset	Description
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP: 1 = Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP: 1 = Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP; 1 = Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP; 1 = Burst on CPU IRQ
23	RO	X	TSENSOR_SLOWDOWN: 0 = Normal; 1 = cpug_clk/2 triggered by the temperature sensor.
16	RO	X	CCLK_RESERVED: Reserved. cpulp uses this bit for div2 bypass.
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0010 = clk_s, 0100 = pllP_out0, 0101 = pllP_out4, 1000 = PLLX_out0 (low jitter), 1001 = dvfs_cpu_clk, 1110 = PLLX_out0, 1111 = dvfs_cpu_clk, (low jitter) 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ

5.2.119 CLK_RST_CONTROLLER_SUPER_CCLKG_DIVIDER_0

Offset: 0x36c | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)
CLK_RST_CONTROLLER_CCLKLP_BURST_POLICY_0

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override. 0 = DISABLE 1 = ENABLE
30	DISABLE	SUPER_CDIV_USE_THERM_CONTROLS: 1 = Use thermal controls for pulse skipper 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.

Bit	Reset	Description
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = COP FIQ does not impact super clock divider enable. 1 = Disable super clock divider on COP FIQ
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = CPU FIQ does not impact super clock divider enable. 1 = Disable super clock divider on CPU FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = COP IRQ does not impact super clock divider enable. 1 = Disable super clock divider on COP IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = CPU IRQ does not impact super clock divider enable. 1 = Disable super clock divider on CPU IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: Divide by [(N/2)+1]
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override.

5.2.120 CLK_RST_CONTROLLER_CCLKLP_BURST_POLICY_0

Offset: 0x370 | Read/Write: R/W | Reset: 0x10X00000 (0b00010000xxxxxxx00000000000000000)

Bit	R/W	Reset	Description
31:28	RW	0x1	CPULP_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP; 1=Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP; 1=Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP; 1=Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP; 1=Burst on CPU IRQ
23	RO	X	RESERVED: Reserved for tsensor_slowdown status for CPUG.
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0010 = clk_s, 0100 = pllP_out0, 0101 = pllP_out4, 1000 = PLLX_out0_LJ, 1001 = DVFS_CPU_CLK, 1110 = PLLX_out0, 1111 = DVFS_CPU_CLK_LJ 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ

Bit	R/W	Reset	Description
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 2 = CLKS 4 = PLLP_OUT0 5 = PLLP_OUT4 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ

5.2.121 CLK_RST_CONTROLLER_SUPER_CCLKLP_DIVIDER_0

Offset: 0x374 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override. 0 = DISABLE 1 = ENABLE
30	DISABLE	SUPER_CDIV_USE_THERM_CONTROLS: 1 = use therm controls for pulse skipper 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = COP FIQ does not impact super clock divider enable. 1 = Disable super clock divider on COP FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = CPU FIQ does not impact super clock divider enable. 1 = Disable super clock divider on CPU FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = COP IRQ does not impact super clock divider enable. 1 = Disable super clock divider on COP IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = CPU IRQ does not impact super clock divider enable. 1 = Disable super clock divider on CPU IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: Divide by [(N/2)+1]
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1. If bit 30 is set, on readback, it gives the final value after hardware (soc_therm) override.

5.2.122 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_0

The CPUG complex consists of the CPUG, L2 cache controller, and a number of bridge devices interfacing the CPUG and L2 cache to the rest of the system. Except for the CPUG, L2 cache controller, and bridge logic to external memory which always runs at the non-divided-down CPUG frequency, other bridge devices can run at various programmable divided-down CPUG clock ratios.

Note: Because the CPUG clock can be selected from 1-of-9 clock sources (refer to the CCLKG_BURST_POLICY register), it is the user’s responsibility to not select a bridge divide-down CPUG clock ratio that will exceed 1/4 of the maximum CPUG frequency supported.

Offset: 0x378 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx0000xxxxxx00)

Bit	Reset	Description
11	0x0	CPUG3_CLK_STP: 1 = CPUG3 clock stop, 0 = CPUG3 clock run.
10	0x0	CPUG2_CLK_STP: 1 = CPUG2 clock stop, 0 = CPUG2 clock run.
9	0x0	CPUG1_CLK_STP: 1 = CPUG1 clock stop, 0 = CPUG1 clock run.
8	0x0	CPUG0_CLK_STP: 1 = CPUG0 clock stop, 0 = CPUG0 clock run.

Bit	Reset	Description
1:0	0x0	CPU_BRIDGE_CLKDIV: Unused but available.

5.2.123 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL_0

Offset: 0x380 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00010000)

Bit	Reset	Description
7:0	0x10	CPU_SOFTRST_LEGACY_WIDTH: CPU soft reset de-assertion counter value for legacy WDT resets.

5.2.124 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL1_0

Offset: 0x384 | Read/Write: R/W | Reset: 0x00040004 (0bxxxx000000000100xxxx000000000100)

Bit	Reset	Description
27:16	0x4	CPU_SOFTRST_DEASSERT_WIDTH: CPU soft reset de-assertion counter value.
11:0	0x4	CPU_SOFTRST_ASSERT_WIDTH: CPU soft reset assertion counter value.

5.2.125 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0

Offset: 0x388 | Read/Write: R/W | Reset: 0x07000200 (0b00xx011100000000xxxx001000000000)

Bit	Reset	Software Default	Description
31	0x0	0x0	IGNORE_HW_ACK_WIDTH: Instructs the flow control state machine to ignore the 16-cpuclk counter and rely only on the sclk counts defined below.
30	0x0	0x0	IGNORE_SW_ACK_WIDTH: Instructs the flow control state machine to ignore the ACK widths below and rely only on 16 cpuclks worth of clamp and reset.
27:16	0x700	0x700	CAR2PMC_NONCPU_ACK_WIDTH: Counter value for ack de-assertion from car2pmc for c0nc/c1nc/crail. Program this field to 1700 cycles of 300M sclk. (~70 cycles of osc_clk-12M)
11:0	0x200	0x0	CAR2PMC_CPU_ACK_WIDTH: Counter value for ack de-assertion from car2pmc for fcpu0/fcpu1/fcpu2/fcpu3/scpu. Program this field to 500 cycles of 300M sclk. (~20 cycles of osc_clk - 12M) Note: Power-On-Reset setting the default reset value of 0x200. For PROD setting, software programs the reset value to is 0x0 for CAR2PMC_CPU_ACK_WIDTH.

5.2.126 CLK_RST_CONTROLLER_CLK_SOURCE_MSELECT_0

Offset: 0x3b4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	MSELECT_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out2, 100 = pllC4_out1, 101 = clk_s, 110 = clk_m, 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT2 4 = PLLC4_OUT1 5 = CLK_S 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	MSELECT_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.127 CLK_RST_CONTROLLER_CLK_SOURCE_TSENSOR_0

Offset: 0x3b8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_S	TSENSOR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 100 = clk_m, 101 = pllC4_out1, 110 = clk_s, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = CLK_M 5 = PLLC4_OUT1 6 = CLK_S 7 = PLLC4_OUT2
7:0	0x0	TSENSOR_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.128 CLK_RST_CONTROLLER_CLK_SOURCE_I2S4_0

Offset: 0x3bc | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	I2S4_CLK_SRC: 000 = pllA_out0, 010 = audio SYNC_CLK 1x or 2x, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S4_MASTER_CLKEN: Reserved.
7:0	0x0	I2S4_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.129 CLK_RST_CONTROLLER_CLK_SOURCE_I2S5_0

Offset: 0x3c0 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	I2S5_CLK_SRC: 000 = pllA_out0, 010 = audio SYNC_CLK 1x or 2x, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 2 = SYNC_CLK 4 = PLLP_OUT0 6 = CLK_M
28	0x1	I2S5_MASTER_CLKEN: Reserved.
7:0	0x0	I2S5_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.130 CLK_RST_CONTROLLER_CLK_SOURCE_I2C4_0

Offset: 0x3c4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
15:0	0x0	I2C4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

5.2.131 CLK_RST_CONTROLLER_CLK_SOURCE_AHUB_0

Offset: 0x3d0 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxx00000xxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	AHUB_CLK_SRC: 000 = pllA_out0, 001 = pllC4_out0, 010 = pllC_out0, 011 = pllC4_out0, 100 = pllP_out0, 101 = pllC4_out2, 110 = clk_m, 111 = ahub_clk_src_alt 0 = PLLA_OUT0 1 = PLLC4_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = PLLP_OUT0 5 = PLLC4_OUT2 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	AHUB_MASTER_CLKEN: Reserved.
20	ENABLE	AHUB_CLK_SRC_DIS: 0 = Enable ahub_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	AHUB_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	AHUB_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.132 CLK_RST_CONTROLLER_CLK_SOURCE_HDA2CODEC_2X_0

Offset: 0x3e4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	HDA2CODEC_2X_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 100 = pllA_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = PLLA_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	HDA2CODEC_2X_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.133 CLK_RST_CONTROLLER_CLK_SOURCE_ACTMON_0

Offset: 0x3e8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	ACTMON_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 100 = clk_s, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = CLK_S 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	ACTMON_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.134 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH1_0

Offset: 0x3ec | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH1_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH1_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.135 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH2_0

Offset: 0x3f0 | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH2_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0 Note: EXTPERIPH2 maps to CLK2
7:0	0x0	EXTPERIPH2_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.136 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH3_0

Offset: 0x3f4 | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH3_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH3_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.137 CLK_RST_CONTROLLER_CLK_SOURCE_I2C_SLOW_0

Offset: 0x3fc | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	I2C_SLOW_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 100 = clk_s, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = CLK_S PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	I2C_SLOW_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.138 CLK_RST_CONTROLLER_CLK_SOURCE_SYS_0

Divider only. All other controls are in SCLK_BURST_POLICY and SUPER_SCLK_DIVIDER.

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SYS_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.139 CLK_RST_CONTROLLER_CLK_SOURCE_SOR1_0

Offset: 0x410 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxx00xxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SOR1_CLK_SRC: 000 = pllP_out0, 010 = pllD_out0, 101 = pllD2_out0, 110 = clk_m, 111 = 1'b0 Non sequential mapping used to preserve 2 MSB to match legacy source select. 0 = PLLP_OUT0 2 = PLLD_OUT0 5 = PLLD2_OUT0 6 = CLK_M
15	0x0	SOR1_CLK_SEL1: 0 = safe clock 24 MHz, 1 = output of SOR1 clock switch
14	0x0	SOR1_CLK_SEL0: 0 = mux output of safe_clk or SOR1 clock switch, 1 = SOR1(DP/HDMI) Brick Output
7:0	0x0	SOR1_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.140 CLK_RST_CONTROLLER_CLK_SOURCE_SOR0_0

Offset: 0x414 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00xxxxxxxxxxxxx)

Bit	Reset	Description
15	0x0	SOR0_CLK_SEL1: 0 = safe clock 24 MHz. Not used.
14	0x0	SOR0_CLK_SEL0: 0 = mux output of safe_clk or LVDS pixel clock, 1 = eDP Macro output

5.2.141 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0

Offset: 0x420 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SATA_OOB_CLK_SRC: 000 = pllP_out0 001 = pllC4_out0 010 = pllC_out0 011 = pllC4_out1 101 = pllC4_out2 110 = clk_m 0 = PLLP_OUT0 1 = PLLC4_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT1 5 = PLLC4_OUT2 6 = CLK_M
7:0	0x0	SATA_OOB_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.142 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0

Offset: 0x424 | Read/Write: R/W | Reset: 0xc1000000 (0b110xxxx1xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SATA_CLK_SRC: 000 = pllP_out0 010 = pllC_out0 011 = pllC4_out0 101 = pllC4_out1 110 = clk_m 111= pllC4_out2 0 = PLLP_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
24	ENABLE	SATA_AUX_CLK_ENB: ENABLE SATA Tx/Rx clocks 0 = DISABLE 1 = ENABLE
7:0	0x0	SATA_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.143 CLK_RST_CONTROLLER_CLK_SOURCE_HDA_0

Offset: 0x428 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	HDA_CLK_SRC: 000 = pllP_out0 001 = pllC2_out0 010 = pllC_out0 011 = pllC4_out0 101 = pllC4_out1 110 = clk_m 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
7:0	0x0	HDA_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.144 CLK_RST_CONTROLLER_RST_DEV_V_SET_0

Offset: 0x430 | Read/Write: R/W | Reset: 0xff81808X (0b111111111xxxxxx11xxxxxxx1xxx0xxx)

Bit	R/W	Reset	Description
29	RW	0x1	SET_HDA_RST: Set reset for the HDA 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_SATA_RST: Set reset for the SATA 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_ACTMON_RST: Set reset for the ACTMON 0 = DISABLE 1 = ENABLE
16	RW	0x1	SET_ATOMICS_RST: Set reset for the ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_HDA2CODEC_2X_RST: Set reset for the HDA2CODEC_2X 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
7	RW	0x1	SET_I2C4_RST: Set reset for the I2C4 0 = DISABLE 1 = ENABLE
3	RW	0x0	SET_MSELECT_RST: Set reset for the MSELECT 0 = DISABLE 1 = ENABLE
1	RO	X	SET_CPULP_RST: Deprecated. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
0	RO	X	SET_CPUG_RST: Deprecated. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.2.145 CLK_RST_CONTROLLER_RST_DEV_V_CLR_0

Offset: 0x434 | Read/Write: R/W | Reset: 0xff81808X (0b111111111xxxxxx11xxxxxxx1xxx0xxx)

Bit	R/W	Reset	Description
29	RW	0x1	CLR_HDA_RST: Clear reset for HDA 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_SATA_RST: Clear reset for SATA 0 = DISABLE 1 = ENABLE
23	RW	0x1	CLR_ACTMON_RST: Clear reset for ACTMON 0 = DISABLE 1 = ENABLE
16	RW	0x1	CLR_ATOMICS_RST: Clear reset for ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_HDA2CODEC_2X_RST: Clear reset for HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_I2C4_RST: Clear reset for I2C4 0 = DISABLE 1 = ENABLE
3	RW	0x1	CLR_MSELECT_RST: Clear reset for MSELECT 0 = DISABLE 1 = ENABLE
1	RO	X	CLR_CPULP_RST: Deprecated. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
0	RO	X	CLR_CPUG_RST: Deprecated. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

5.2.146 CLK_RST_CONTROLLER_RST_DEV_W_SET_0

Offset: 0x438 | Read/Write: R/W | Reset: 0x190077ff (0bxxx11xx1xx0xxxxx111x11111111111)

Bit	Reset	Description
28	0x1	SET_XUSB_SS_RST: Set reset for the XUSB_SS logic 0 = DISABLE 1 = ENABLE
27	0x1	SET_DVFS_RST: Set reset for the CLDVFS 0 = DISABLE 1 = ENABLE
21	0x0	SET_ENTROPY_RST: Set reset for the ENTROPY logic. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	0x1	SET_XUSB_PADCTL_RST: Set reset for the XUSB_PADCTL logic. 0 = DISABLE 1 = ENABLE
13	0x1	SET_RESERVED10_RST: Reserved.
12	0x1	SET_RESERVED9_RST: Reserved.
10	0x1	SET_RESERVED7_RST: Reserved.
9	0x1	SET_RESERVED6_RST: Reserved.
8	0x1	SET_CEC_RST: Set the reset for the CEC 0 = DISABLE 1 = ENABLE
7	0x1	SET_RESERVED5_RST: Reserved.
6	0x1	SET_RESERVED4_RST: Reserved.
5	0x1	SET_RESERVED3_RST: Reserved.
4	0x1	SET_RESERVED2_RST: Reserved.
3	0x1	SET_RESERVED1_RST: Reserved.
2	0x1	SET_RESERVED0_RST: Reserved.
1	0x1	SET_SATACOLD_RST: Set reset for the SATACOLD 0 = DISABLE 1 = ENABLE
0	0x1	SET_HDA2HDMICODEC_RST: Set reset for the HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

5.2.147 CLK_RST_CONTROLLER_RST_DEV_W_CLR_0

Offset: 0x43c │ Read/Write: R/W │ Reset: 0x190077ff (0bxxx11xx1xx0xxxxxx111x11111111111)

Bit	R/W	Reset	Description
28	RW	0x1	CLR_XUSB_SS_RST: Clear reset for the XUSB_SS logic. 0 = DISABLE 1 = ENABLE
27	RW	0x1	CLR_DVFS_RST: Clear reset for CLDVFS 0 = DISABLE 1 = ENABLE
21	RW	0x0	CLR_ENTROPY_RST: Clear reset for the ENTROPY logic 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_XUSB_PADCTL_RST: Clear reset for XUSB_PADCTL logic 0 = DISABLE 1 = ENABLE
13	RW	0x1	CLR_RESERVED10_RST: Reserved.
12	RW	0x1	CLR_RESERVED9_RST: Reserved.
10	RW	0x1	CLR_RESERVED7_RST: Reserved.
9	RW	0x1	CLR_RESERVED6_RST: Reserved.
8	RW	0x1	CLR_CEC_RST: Clear reset for CEC 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_RESERVED5_RST: Reserved.
6	RW	0x1	CLR_RESERVED4_RST: Reserved.
5	RW	0x1	CLR_RESERVED3_RST: Reserved.
4	RW	0x1	CLR_RESERVED2_RST: Reserved.
3	RW	0x1	CLR_RESERVED1_RST: Reserved.
2	RW	0x1	CLR_RESERVED0_RST: Reserved.

Bit	R/W	Reset	Description
1	RW	0x1	CLR_SATACOLD_RST: Clear reset for SATACOLD 0 = DISABLE 1 = ENABLE
0	RW	0x1	CLR_HDA2HDMICODEC_RST: Clear reset for HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

5.2.148 CLK_RST_CONTROLLER_CLK_ENB_V_SET_0

Offset: 0x440 | Read/Write: R/W | Reset: 0x00400008 (0b0000000001xxxxx00xxx00xx00001x00)

Bit	Reset	Description
29	0x0	SET_CLK_ENB_HDA: Set enable clock to HDA 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_SATA: Set enable clock to SATA 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_SATA_OOB: Set enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_EXTPERIPH3: Set enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_EXTPERIPH2: Set enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	0x0	SET_CLK_ENB_EXTPERIPH1: Set enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_ACTMON: Set enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	0x1	SET_CLK_ENB_SPDIF_DOUBLER: Set enable clock to S/PDIF audio sync doubler 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_ATOMICS: Set enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_HDA2CODEC_2X: Set enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_APB2APE: Set enable APB clock to APE 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_AHUB: Set enable clock to AHUB 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_I2C4: Set enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_I2S5: Set enable clock to I2S5 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_I2S4: Set enable clock to I2S4 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_TSENSOR: Set enable clock to TSENSOR 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x1	SET_CLK_ENB_MSELECT: Set enable clock to MSELECT 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_CPUG: Set enable clock to CPUG. 0 = DISABLE 1 = ENABLE

5.2.149 CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00400008 (0b0000000001xxxxx00xxx00xx00001x00)

Bit	Reset	Description
29	0x0	CLR_CLK_ENB_HDA: Clear enable clock to HDA 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_SATA: Clear enable clock to SATA 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_SATA_OOB: Clear enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_EXTPERIPH3: Clear enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_EXTPERIPH2: Clear enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	0x0	CLR_CLK_ENB_EXTPERIPH1: Clear enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_ACTMON: Clear enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	0x1	CLR_CLK_ENB_SPDIF_DOUBLER: Clear enable clock to S/PDIF audio sync doubler 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_ATOMICS: Clear enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_HDA2CODEC_2X: Clear enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_APB2APE: Clear enable APB clock to APE 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_AHUB: Clear enable clock to AHUB 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_I2C4: Clear enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_I2S5: Clear enable clock to I2S5 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_I2S4: Clear enable clock to I2S4 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_TSENSOR: Clear enable clock to TSENSOR 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x1	CLR_CLK_ENB_MSELECT: Clear enable clock to MSELECT 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_CPUG: Clear enable clock to CPUG. 0 = DISABLE 1 = ENABLE

5.2.150 CLK_RST_CONTROLLER_CLK_ENB_W_SET_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x402000fc (0bx1000xx0xx1000000x00x00011111100)

Bit	Reset	Description
30	0x1	SET_CLK_ENB_ MC1: Set enable clock to MC1 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_EMC_DLL: Set enable clock to EMC_DLL 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_XUSB_SS: Set enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_DVFS: Set enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_ENTROPY: Set the enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_DSIB_LP: Set the enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CLK_ENB_DSIA_LP: Set the enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_CILEF: Set the enable clock to CSI CILEF. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_CILCD: Set the enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_CILAB: Set the enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_XUSB: Set the enable clock to XUSB 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_MIPI_IOBIST: Set the enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_SATA_IOBIST: Set the enable clock to SATA IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_EMC_IOBIST: Set the enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_PCIE2_IOBIST: Set the enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_CEC: Set the enable clock to CEC 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x1	SET_CLK_ENB_PCIERX5: Set the enable clock to PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	SET_CLK_ENB_PCIERX4: Set the enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	SET_CLK_ENB_PCIERX3: Set the enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	SET_CLK_ENB_PCIERX2: Set the enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	SET_CLK_ENB_PCIERX1: Set the enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	SET_CLK_ENB_PCIERX0: Set the enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_RESERVED0: Reserved SATACOLD 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_HDA2HDMICODEC: Set the enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

5.2.151 CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0

Offset: 0x44c │ Read/Write: R/W │ Reset: 0x402000fc (0bx1000xx0xx1000000x00x00011111100)

Bit	Reset	Description
30	0x1	CLR_CLK_ENB_MC1: Clear the enable clock to MC1 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_EMC_LATENCY: Clear the enable clock to EMC_LATENCY 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_XUSB_SS: Clear the enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_DVFS: Clear the enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_ENTROPY: Clear the enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_DSIB_LP: Clear the enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_DSIA_LP: Clear the enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_CILEF: Clear the enable clock to CSI CILEF. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_CILCD: Clear the enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_CILAB: Clear the enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x0	CLR_CLK_ENB_XUSB: Clear the enable clock to XUSB 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_MIPI_IOBIST: Clear the enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SATA_IOBIST: Clear the enable clock to SATA IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_EMC_IOBIST: Clear the enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_PCIE2_IOBIST: Clear the enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_CEC: Clear the enable clock to CEC 0 = DISABLE 1 = ENABLE
7	0x1	CLR_CLK_ENB_PCIERX5: Clear the enable clock to PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	CLR_CLK_ENB_PCIERX4: Clear the enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	CLR_CLK_ENB_PCIERX3: Clear the enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	CLR_CLK_ENB_PCIERX2: Clear the enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	CLR_CLK_ENB_PCIERX1: Clear the enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CLK_ENB_PCIERX0: Clear the enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_RESERVED0: Reserved.
0	0x0	CLR_CLK_ENB_HDA2HDMICODEC: Clear the enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

5.2.152 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_SET_0

Offset: 0x450 | Read/Write: R/W | Reset: 0x2000feef (0bx010xxx00000000011111111011101111)

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	SET_NONCPURESET: 1 = Assert reset to the whole nonCPU region of the CPU. For FCCPLEX, by default the NONCPU region is not power-gated but the reset is asserted because CRAIL is power-gated by default. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: Reserved.
24	0x0	SET_L2RESET: 1 = Assert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CXRESET3: Reserved
22	0x0	SET_CXRESET2: Reserved
21	0x0	SET_CXRESET1: Reserved

Bit	Reset	Description
20	0x0	SET_CXRESET0: Reserved
19	0x0	SET_CORERESET3: 1 = Assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CORERESET2: 1 = Assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CORERESET1: 1 = Assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CORERESET0: 1 = Assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	SET_DBGRESET3: Reserved
14	0x1	SET_DBGRESET2: Reserved
13	0x1	SET_DBGRESET1: Reserved
12	0x1	SET_DBGRESET0: Reserved
11	0x1	SET_WDRESET3: Reserved.
10	0x1	SET_WDRESET2: Reserved.
9	0x1	SET_WDRESET1: Reserved.
8	0x0	SET_WDRESET0: Reserved.
7	0x1	SET_DERESET3: Reserved.
6	0x1	SET_DERESET2: Reserved.
5	0x1	SET_DERESET1: Reserved.
4	0x0	SET_DERESET0: Reserved.
3	0x1	SET_CPURESET3: 1 = Assert nCPUPORESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = Assert nCPUPORESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPURESET1: 1 = Assert nCPUPORESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	SET_CPURESET0: 1 = Assert nCPUPORESET to CPU0. 0 = DISABLE 1 = ENABLE

5.2.153 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_CLR_0

Offset: 0x454 | Read/Write: R/W | Reset: 0x2000feef (0bx010xxx00000000011111111011101111)

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = De-assert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	CLR_NONCPURESET: 1 = De-assert reset to the whole nonCPU region of the CPU. For FCCPLEX, by default the NONCPU region is not power-gated but the reset is asserted because CRAIL is power-gated by default. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: Reserved.
24	0x0	CLR_L2RESET: 1 = De-assert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CXRESET3: Reserved
22	0x0	CLR_CXRESET2: Reserved

Bit	Reset	Description
21	0x0	CLR_CXRESET1: Reserved
20	0x0	CLR_CXRESET0: Reserved
19	0x0	CLR_CORERESET3: 1 = De-assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CORERESET2: 1 = De-assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CORERESET1: 1 = De-assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CORERESET0: 1 = De-assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DBGRESET3: Reserved
14	0x1	CLR_DBGRESET2: Reserved
13	0x1	CLR_DBGRESET1: Reserved
12	0x1	CLR_DBGRESET0: Reserved
11	0x1	CLR_WDRESET3: Reserved.
10	0x1	CLR_WDRESET2: Reserved.
9	0x1	CLR_WDRESET1: Reserved.
8	0x0	CLR_WDRESET0: Reserved.
7	0x1	CLR_DERESET3: Reserved.
6	0x1	CLR_DERESET2: Reserved.
5	0x1	CLR_DERESET1: Reserved.
4	0x0	CLR_DERESET0: Reserved.
3	0x1	CLR_CPURESET3: 1 = De-assert nCPUPORESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CPURESET2: 1 = De-assert nCPUPORESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_CPURESET1: 1 = De-assert nCPUPORESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_CPURESET0: 1 = De-assert nCPUPORESET to CPU0. 0 = DISABLE 1 = ENABLE

5.2.154 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_SET_0

Offset: 0x460 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

Bit	Reset	Description
11	0x0	SET_CPU3_CLK_STP: 1 = Assert CPU3 clock stop 0 = DISABLE 1 = ENABLE
10	0x0	SET_CPU2_CLK_STP: 1 = Assert CPU2 clock stop 0 = DISABLE 1 = ENABLE
9	0x0	SET_CPU1_CLK_STP: 1 = Assert CPU1 clock stop 0 = DISABLE 1 = ENABLE
8	0x0	SET_CPU0_CLK_STP: 1 = Assert CPU0 clock stop 0 = DISABLE 1 = ENABLE

5.2.155 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_CLR_0

Offset: 0x464 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

Bit	Reset	Description
11	0x0	CLR_CPU3_CLK_STP: 1 = De-assert CPU3 clock stop 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CPU2_CLK_STP: 1 = De-assert CPU2 clock stop 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CPU1_CLK_STP: 1 = De-assert CPU1 clock stop 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CPU0_CLK_STP: 1 = De-assert CPU0 clock stop 0 = DISABLE 1 = ENABLE

5.2.156 CLK_RST_CONTROLLER_CPU_CMPLX_STATUS_0

Offset: 0x470 | Read/Write: RO | Reset: 0xXXXXXX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	PRESETDBG: ENABLE = nPRESETDBG asserted to the CoreSight. 0 = DISABLE 1 = ENABLE
29	X	NONCPURESET: ENABLE = Reset asserted to the whole nonCPU region of the CPU 0 = DISABLE 1 = ENABLE
28	X	MCRESET: ENABLE = mcreset_ asserted 0 = DISABLE 1 = ENABLE
27	X	CSITEPTMRESET: ENABLE = nCSITEPTMRESET asserted 0 = DISABLE 1 = ENABLE
26	X	AXICIFRESET: ENABLE = nAXICIFRESET asserted 0 = DISABLE 1 = ENABLE
25	X	MPSELECTRESET: ENABLE = mselectreset asserted 0 = DISABLE 1 = ENABLE
24	X	L2RESET: ENABLE = L2RESET asserted to the CPU 0 = DISABLE 1 = ENABLE
23	X	CXRESET3: Reserved
22	X	CXRESET2: Reserved
21	X	CXRESET1: Reserved
20	X	CXRESET0: Reserved
19	X	CORERESET3: ENABLE = nCORERESET asserted to CPU3 0 = DISABLE 1 = ENABLE
18	X	CORERESET2: ENABLE = nCORERESET asserted to CPU2 0 = DISABLE 1 = ENABLE
17	X	CORERESET1: ENABLE = nCORERESET asserted to CPU1 0 = DISABLE 1 = ENABLE
16	X	CORERESET0: ENABLE = nCORERESET asserted to CPU0 0 = DISABLE 1 = ENABLE
15	X	DBGRESET3: Reserved
14	X	DBGRESET2: Reserved

Bit	Reset	Description
13	X	DBGRESET1: Reserved
12	X	DBGRESET0: Reserved
11	X	WDRESET3: Reserved.
10	X	WDRESET2: Reserved.
9	X	WDRESET1: Reserved.
8	X	WDRESET0: Reserved.
3	X	CPURESET3: ENABLE = nCPUPORESET asserted to CPU3 0 = DISABLE 1 = ENABLE
2	X	CPURESET2: ENABLE = nCPUPORESET asserted to CPU2 0 = DISABLE 1 = ENABLE
1	X	CPURESET1: ENABLE = nCPUPORESET asserted to CPU1 0 = DISABLE 1 = ENABLE
0	X	CPURESET0: ENABLE = nCPUPORESET asserted to CPU0 0 = DISABLE 1 = ENABLE

5.2.157 CLK_RST_CONTROLLER_INTSTATUS_0

Interrupt Status Register.

- car_int[0] = axicifreset
- car_int[1] = csiteptmreset
- car_int[2] = periphreset
- car_int[3] = scureset
- car_int[4] = cpureset_cpu0
- car_int[5] = cpureset_cpu1
- car_int[6] = cpureset_cpu2
- car_int[7] = cpureset_cpu3
- car_int[8] = dbgreset_cpu0
- car_int[9] = dbgreset_cpu1
- car_int[10] = dbgreset_cpu2
- car_int[11] = dbgreset_cpu3
- car_int[12] = wdreset_cpu0
- car_int[13] = wdreset_cpu1
- car_int[14] = wdreset_cpu2
- car_int[15] = wdreset_cpu3
- car_int[16] = tsensor2car_slowdown_sclk
- car_int[17] = spare
- car_int[18] = spare
- car_int[19] = spare

Offset: 0x478 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	CAR_INT: Clear on 1-write. The initial value is cleared.

5.2.158 CLK_RST_CONTROLLER_INTMASK_0

Interrupt Mask Register.

Offset: 0x47c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	CAR_INTMASK: Mask 0=masked, 1=unmasked. Init masked

5.2.159 CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0

The data sampling frequency relies on a 960 MHz clock, so the goal of the PLL is to have:
In_Frequency * (PLL_VCOMULTBY2+1) * (PLL_NDIV/PLL_MDIV) = 960 MHz. With a 12 MHz input from PLL_U, the default setting of PLL_VCOMULTBY2 = 1, PLL_NDIV = 40, and PLL_MDIV = 1 results in a correct output.

This register is used to configure the PHY PLL contained in the UTMIP module.

UTMIP PLL Configuration Register 0

Offset: 0x480 | Read/Write: R/W | Reset: 0x001901XX (0bxxx00000000110010000000 1xxxxx001)

Bit	R/W	Reset	Description
28:27	RW	0x0	UTMIP_PLL_VREG_CTRL: Voltage regulator voltage level control.
26:25	RW	0x0	UTMIP_PLL_KCP: KCP of the UTMIP PHY PLL. Charge Pump Gain control. Default value is zero. See cell specification.
23:16	RW	0x19	UTMIP_PLL_NDIV: NDIV[7:0] input of UTMIP PLL. This is the feedback divider on the VCO feedback. 0x0 is not allowed. See cell specification.
15:8	RW	0x1	UTMIP_PLL_MDIV: MDIV[7:0] input of the UTMIP PLL. This is the predivide on the PLL. 0x0 is not allowed. See cell specification.
3	RO	X	UTMIP_PLL_RESERVED: Reserved.
2	RW	0x0	UTMIP_PLL_LOCK_OVR: LOCK_OVERRIDE control of the UTMIP PLL. Forces PLL_LOCK to 1. See cell specification.
1	RW	0x0	UTMIP_PLL_RESERVED2: RESERVED.
0	RW	0x1	UTMIP_PLL_EN_LCKDET: ENB_LCKDET input of UTMIP PLL. When set to 1, powers down the lock detect. Setting ENB_LCKDET=X, LOCK_OVERRIDE=1, PLL_LOCK=1, and PLL_FREQLOCK=1 also powers down lock detect. 0 0 0->1 (after lock) 0->1 (after lock) 1 0 0 0 lock detect power down.

5.2.160 CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0

UTMIP PLL Configuration and Parameters

In normal operation, the following clock generators are in play for USB:

Crystal (Xtal) clock -> enters PLL_U to generate 12 MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock.

The following parameters control the bring-up of the PLLs (coming out of reset or suspend):

- PllUOnState: start pllu_enable_count and pll_lock_count
  - Wait ~1 µs to enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)
  - Wait ~1 ms until PLL_U is stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE
  - pll_active_count = 0

- Wait 10 µs to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE

- Wait ~2.5 ms until USB_PHY is stable (pll_lock_count == ClkXtal * XTAL_FREQ_COUNT * 256) => USB_PHY

Values for a 19.2 MHz Xtal clock are:

- PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 19.2 / 8 = 2.4 = 3 = 0x03
- PLLU_STABLE_COUNT[11:0] = 1000 * 19.2 / 256 = 75 = 0x4b
- PLL_ACTIVE_DLY_COUNT[5:0] = 10 * 19.2 / 16 = 12 = 0x0c
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

UTMIP PLL and PLLU Configuration Register 1

Offset: 0x484 | Read/Write: R/W | Reset: 0x180150c0 (0b00011000000000001010100000011000000)

Bit	Reset	Software Default	Description
31:27	0x3	0x0	UTMIP_PLLU_ENABLE_DLY_COUNT: Controls the wait time to enable PLL_U when coming out of suspend or reset.
26:18	0x0	NONE	UTMIP_PLL_RSVD: Reserved
17	0x0	NONE	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
16	0x1	NONE	UTMIP_FORCE_PLLU_POWERDOWN: Force PLL_U into power down. (Overrides FORCE_PLLU_POWERUP.)
15	0x0	NONE	UTMIP_FORCE_PLL_ENABLE_POWERUP: Force UTMIP PLL pll_enable input on.
14	0x1	NONE	UTMIP_FORCE_PLL_ENABLE_POWERDOWN: Force UTMIP PLL pll_enable input off. (Overrides FORCE_PLL_ENABLE_POWERUP.)
13	0x0	NONE	UTMIP_FORCE_PLL_ACTIVE_POWERUP: Force UTMIP PLL pll_active input on.
12	0x1	NONE	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Force UTMIP PLL pll_active input off. (Overrides FORCE_PLL_ACTIVE_POWERUP.
11:0	0xc0	0x80	UTMIP_XTAL_FREQ_COUNT: Determines the time to wait until the output of UTMIP PLL is considered stable.

5.2.161 CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0

UTMIP Miscellaneous Configurations

Offset: 0x488 | Read/Write: R/W | Reset: 0xXX318015 (0bx1xxxx01001100011000000000010101)

Bit	R/W	Reset	Software Default	Description
30	RW	0x1	NONE	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29:26	RO	X	NONE	UTMIP_FORCE_PD_RESERVED: Reserved.
25	RW	0x0	NONE	UTMIP_FORCE_PD_SAMP_D_POWERUP: Force UTMIP PLL PD_SAMP_D input (XUSB DEV) into power up.
24	RW	0x1	NONE	UTMIP_FORCE_PD_SAMP_D_POWERDOWN: Force UTMIP PLL PD_SAMP_D input (XUSB DEV) into power down. (Overrides FORCE_PD_SAMP_D_POWERUP.)
23:18	RW	0xc	NONE	UTMIP_PLL_ACTIVE_DLY_COUNT: 10 µs / (1/19.2MHz) = 192 / 16 = 12
17:6	RW	0x600	0x0	UTMIP_PLLU_STABLE_COUNT: PLLU frequency lock delay (See comments above on correct delay and calculation)
5	RW	0x0	NONE	UTMIP_FORCE_PD_SAMP_C_POWERUP: Force UTMIP PLL PD_SAMP_C input into power up.

Bit	R/W	Reset	Software Default	Description
4	RW	0x1	NONE	UTMIP_FORCE_PD_SAMP_C_POWERDOWN: Force UTMIP PLL PD_SAMP_C input into power down. (Overrides FORCE_PD_SAMP_C_POWERUP.)
3	RW	0x0	NONE	UTMIP_FORCE_PD_SAMP_B_POWERUP: Force UTMIP PLL PD_SAMP_B input (XUSB HOST) into power up.
2	RW	0x1	NONE	UTMIP_FORCE_PD_SAMP_B_POWERDOWN: Force UTMIP PLL PD_SAMP_B input (XUSB HOST) into power down. (Overrides FORCE_PD_SAMP_B_POWERUP.)
1	RW	0x0	NONE	UTMIP_FORCE_PD_SAMP_A_POWERUP: Force UTMIP PLL PD_SAMP_A input into power up.
0	RW	0x1	NONE	UTMIP_FORCE_PD_SAMP_A_POWERDOWN: Force UTMIP PLL PD_SAMP_A input into power down. (Overrides FORCE_PD_SAMP_A_POWERUP.)

5.2.162 CLK_RST_CONTROLLER_PLLE_AUX_0

The range value is the recommended range. All counters support 0-255 step range.

Offset: 0x48c │ Read/Write: R/W │ Reset: 0x0X002070 (0b0xx0xx10000000000010000001110000)

Bit	R/W	Reset	Software Default	Description
31	RW	SKIP	0x1	PLLE_SS_SEQ_INCLUDE: 0=Skip, 1=Include. Include PLLE spread spectrum power sequencer. If this bit needs to be toggled, do it before writing '1' to PLLE_SEQ_ENABLE. 0 = SKIP 1 = INCLUDE
28	RW	0x0	NONE	PLLE_REF_SEL_PLLREFE: PLLE input reference clock source select2. 0 = Select the setting of the PLLE_REF_SRC field mentioned below, 1 = Select PLLREFE_out which is 600M (use pre-divM=50 for 12 MHz reference)
27:26	RO	X	NONE	PLLE_SEQ_STATE: PLLE power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	NONE	PLLE_SEQ_START_STATE: PLLE power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	0x1	PLLE_SEQ_ENABLE: PLLE power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
23:16	RW	0x0	NONE	PLLE_SS_DLY: PT6: Delay between spread spectrum ON<->OFF transitions in SS power toggle sequence: ON->OFF->ON. Range is 0-255 ms in 1 ms steps.
15:8	RW	0x20	NONE	PLLE_LOCK_DLY: PT5: Delay from PLLE ENABLE to lock. Range is 0-128 µs in 1 µs steps
7	RW	0x0	NONE	TEST_FAST_PT: 0=Normal timer steps. 1=Fast timer steps. Fast programmable timer steps size for testing. Normal is per µs or ms, fast is per oscillator clock. Applies to all programmable timer counters in SATA, PCIe and PLLE seq.
6	RW	0x1	0x0	PLLE_SS_SWCTL: 0=PLLE spread spectrum configuration setup by hardware, 1=Setup by software during training
5	RW	0x1	NONE	PLLE_CONFIG_SWCTL: This bit should never be written to 0. Do not change the setup bits once written by software. 0=PLLE config setup by hardware, 1=PLLE setup by software during training. Affects resettable bits when PLL is off (SETUP and EXT_SETUP bits). All other config bits are left untouched as initialized, for example, M/N/P, SS coefficient.
4	RW	0x1	0x0	PLLE_ENABLE_SWCTL: 0=PLLE enable by hardware, 1=PLLE enable by software

3	RW	0x0	0x1	PLLE_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLLE
2	RW	OSC_DIV	NONE	PLLE_REF_SRC: PLLE input reference clock source select. 0=OSC_DIV (options: mux (PLLs, ext_osc), /2, /4), 1=PLLP_OUT0 (use pre-divM=18 for 12 MHz reference) 0 = OSC_DIV 1 = PLLP_OUT0
1	RW	DISABLE	NONE	PLLE_CML1_OEN: PLLE CML1 clock out enable. Used for SATA. 0 = DISABLE 1 = ENABLE
0	RW	DISABLE	NONE	PLLE_CML0_OEN: PLLE CML0 clock out enable. Used for PCIe. 0 = DISABLE 1 = ENABLE

5.2.163 CLK_RST_CONTROLLER_SATA_PLL_CFG0_0

SATA Power Sequencer Input Software Control Program Guide

The software control input can be used for test coverage or diving the power sequencer SM via software. The SATA power sequencer is driven by 3 primary reset and power-down input signals: reset, lane_pd, and padpll_pd. The normal power-up sequence de-asserts in this order: padpll_pd > lane_pd > reset. The normal power-down sequence asserts in this order: reset > lane_pd > padpll_pd.

It is acceptable to assert all three signals at the same time. It is also acceptable for reset or lane_pd to return to the power-up state in the middle of a power-down sequence.

Offset: 0x490 │ Read/Write: R/W │ Reset: 0x0X0X3803 (0b0000xx10xxxxxxxx001110000000x011)

Bit	R/W	Reset	Software Default	Description
31:28	RW	0x0	NONE	SATA_PADPLL_IDDQ2_PADPLL_RCAL_DLY: PT1: Delay from SATAX PAD PLL IDDQ DE assertion to the PAD PLL RCAL EN assertion. Should be at least 50ns and FSM delays take care of it.
27:26	RO	X	NONE	SATA_SEQ_STATE: SATA power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	NONE	SATA_SEQ_START_STATE: SATA power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	0x1	SATA_SEQ_ENABLE: SATA power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
16	RO	X	NONE	SATA_PADPLL_CAL_VALID: Indicator that Brick PLL calibration was done and is valid. CAL_VALID_RST bit should make this '0'.
15	RW	0x0	NONE	SATA_PADPLL_REQ_PLL_RCAL_BYPASS: 0= Don’t bypass Resistor calibration 1=BYPASS Resistor calibration
14	RW	0x0	NONE	SATA_PADPLL_REQ_PLL_RCAL: a trigger signal needs a rising edge on this bit to Force PLL Resistor Calibration
13	RW	0x1	NONE	SATA_PADPLL_SLEEP_IDDQ: IDDQ value during SLEEP mode
12:11	RW	0x3	NONE	SATA_PADPLL_SLEEP_VAL: select type of PLL sleep states from 1 to 3
10	RW	0x0	NONE	SATA_PADPLL_CAL_VALID_RST: 0= do not reset 1=reset the CAL VALID
9	RW	0x0	NONE	SATA_PADPLL_REQ_PLL_CAL_RESET: a trigger signal needs a rising edge on this bit to reset PLL VCO calibration

Bit	R/W	Reset	Software Default	Description
7	RW	0x0	NONE	SATA_SEQ_PADPLL_PD_INPUT_VALUE: 0=Pad PLL is powered up 1=Software control pad PLL powered down (PD) by setting this and SATA_SEQ_IN_SWCTL bit
6	RW	0x0	NONE	SATA_SEQ_LANE_PD_INPUT_VALUE: 0=I/O PHY is powered up 1=Software control I/O PHY powered down by setting this and SATA_SEQ_IN_SWCTL bit
5	RW	0x0	NONE	SATA_SEQ_RESET_INPUT_VALUE: 0=SATA reset de-asserted, 1=Software control reset by setting this and SATA_SEQ_IN_SWCTL bit
4	RW	0x0	NONE	SATA_SEQ_IN_SWCTL: 0=Seq input by hardware 1=Seq input by software
2	RW	0x0	0x1	SATA_PADPLL_USE_LOCKDET: 0=Use programmable delay 1=Use lockdet signals from PLL
1	RW	0x1	NONE	SATA_PADPLL_RESET_OVERRIDE_VALUE: 0=Pad PLL is powered up 1=Software control reset by setting this and SATA_PADPLL_RESET_SWCTL bit
0	RW	0x1	0x0	SATA_PADPLL_RESET_SWCTL: 0=Reset by hardware 1=Reset by software

5.2.164 CLK_RST_CONTROLLER_SATA_PLL_CFG1_0

The range value is the recommended range. All counters support 0-255 µs range in 1 µs steps

Offset: 0x494 │ Read/Write: R/W │ Reset: 0x00011400 (0b00000000000000010001010000000000)

Bit	Reset	Description
31:24	0x0	SATA_LANE_IDDQ2_PADPLL_RESET_DLY: PT4: Delay from placing lane out of IDDQ to PAD PLL out of reset. Range is 0-200 µs.
23:16	0x1	SATA_PADPLL_IDDQ2LANE_SLUMBER_DLY: PT3: Delay from SATA PAD PLL out of IDDQ to lane placed out of IDDQ. Range is 0-1 µs.
15:8	0x14	SATA_PADPLL_PU_POST_DLY: PT2: Delay from ENABLE to SATA PAD PLL lock. Range is 1-20 µs.
7:0	0x0	SATA_LANE_IDDQ2_PADPLL_IDDQ_DLY: PT1: Delay from placing lane in IDDQ to PAD PLL in IDDQ.

5.2.165 CLK_RST_CONTROLLER_PCIE_PLL_CFG_0

PCIe Power Sequencer Input Software Control Program Guide

The software control input can be used for test coverage or driving power sequencer SM via software.

The PCIe power sequencer is driven by 1 primary reset input signal: reset. Reset is expected to be held static until the power-up or power-down sequence is complete.

Offset: 0x498 │ Read/Write: R/W │ Reset: 0x0X00000c (0bxxxxxx10xxxxxxxxxxxxxxxxxx0011xx)

Bit	R/W	Reset	Description
27:26	RO	X	PCIE_SEQ_STATE: PCIe power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY

Bit	R/W	Reset	Description
25	RW	SEQ_ON	PCIE_SEQ_START_STATE: PCIe power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	PCIE_SEQ_ENABLE: PCIe power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
5	RW	0x0	PCIE_SEQ_RESET_INPUT_VALUE: 0=PCIe reset de-asserted, 1=Software control reset by setting this and PCIE_SEQ_IN_SWCTL bit
4	RW	0x0	PCIE_SEQ_IN_SWCTL: 0=Sequencer input by hardware, 1=Sequencer input by software
3	RW	ENABLE	PCIE_XCLK_ENABLE_OVERRIDE_VALUE: Override value used only when PCIE_XCLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	PCIE_XCLK_ENABLE_SWCTL: 0=XCLK enabled by hardware, 1=XCLK enabled by software

5.2.166 CLK_RST_CONTROLLER_PROG_AUDIO_DLY_CLK_0

- Clock doubler with 1X/2X is available for the I2S1-I2S5 and S/PDIF clock.
- The CLK_ENB_*_DOUBLE bits in the CLK_OUT_ENB_V register must be enabled for either SYNC_1X_CLK or SYNC_2X_CLK to operate.
- PROG_DLY_CLK_SPDIF provides programmable delay for the clock doublers.
- Audio sync clock source mux options: SPDIFIN, I2S1, I2S2, I2S3, I2S4, I2S5, PLLA_OUT0, EXT_VIMCLK

Enable Sync Clocks

There are 3 related clock enables/selects to enable the sync clocks:

- CG_1. AUDIO_SYNC_CLK_I2S1-SYNC_CLK_DIS: disable output of sync_clk mux
- CG_2. CLK_OUT_ENB_V-CLK_ENB_I2S1_DOUBLER: disable input to doubler
- SEL_3. PROG_AUDIO_DLY_CLK-I2S1_1X_SEL: Select 1x sync clock. 0=2x, 1=1x

Flow of Clock Logic

sync_clk mux (CG_1) -> (CG_2) (SEL_3) doubler -> clk switch

Suggested Programming for Sync Clock Enable:

- Leave CG_2 as default enabled.
- Use CG_1 to disable or enable (default) sync clock.
- Use SEL_3 to select 1x or 2x sync clock.

SYNC_CLK_DIS	CLK_ENB_I2S1_DOUBLER	I2S1_1X_SEL	SYNC_CLK
0	1	0	2x (Default)
0	1	1	1x
1	1	X	0 (off, recommended)
1	X	X	0 (off)
X	0	X	0 (off)

Offset: 0x49c │ Read/Write: R/W │ Reset: 0x00700000 (0bxx0xxxxx0111xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	0x0	SPDIF_1X_SEL: Select S/PDIF 1x sync_clk. 0=sync_2x_clk, 1=sync_clk

23:20	0x7	SYNC_CLK_SPDIF_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler

5.2.167 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S1_0

Audio Sync Clock Source Select

Offset: 0x4a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock.
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.168 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S2_0

Audio Sync Clock Source Select

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = reserved 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.169 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S3_0

Audio Sync Clock Source Select

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.170 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S4_0

Audio Sync Clock Source Select

Offset: 0x4ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.171 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S5_0

Audio Sync Clock Source Select

Offset: 0x4b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.172 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_SPDIF_0

Audio Sync Clock Source Select

Offset: 0x4b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.173 CLK_RST_CONTROLLER_PLLD2_BASE_0

Offset: 0x4b8 | Read/Write: R/W | Reset: 0xXX041401 (0b000xx000000001x00001010000000001)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD2_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD2_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD2_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
28	RO	X	PLLD2_FREQLOCK: 0 = not lock, 1 = lock frequency
27	RO	X	PLLD2_LOCK: 0 = not lock, 1 = lock.
26:25	RW	0x0	PLLD2_REF_SRC_SEL: Reference source select sel[0]=0 && sel[1]=1 => ref_src = pllP_out0 sel[0]=0 && sel[1]=0 => ref_src = osc_div_clk 0 = PLL_D 1 = PLL_D2
24	RW	0x0	PLLD2_LOCK_OVERRIDE:Forces PLL_LOCK to 1.
23:19	RW	0x0	PLLD2_PDIV: P divider.
18	RW	0x1	PLLD2_IDDQ: 0 The PLL is powered up 1: Software can put the PLL in IDDQ by setting this bit 0 = OFF 1 = ON
16	RW	0x0	PLLD2_PTS: Base PLLD2 test output select. 0 = PTO is 0 1 = PTO is FO 0 = DISABLE 1 = FO
15:8	RW	0x14	PLLD2_NDIV: N divider.
7:0	RW	0x1	PLLD2_MDIV: M divider.

5.2.174 CLK_RST_CONTROLLER_PLLD2_MISC_0

Offset: 0x4bc | Read/Write: R/W | Reset: 0x40000000 (0bx1xxx00000000000000000000000000)

Bit	Reset	Software Default	Description
30	ENABLE	NONE	PLLD2_EN_LCKDET: 0 = DISABLE 1 = ENABLE
26:25	0x0	NONE	PLLD2_ KCP: Charge pump gain control.
24	0x0	NONE	PLLD2_ KVCO:VCO gain.
23:0	0x0	0x20	PLLD2_ SETUP: setup[23:0].

5.2.175 CLK_RST_CONTROLLER_UTMIP_PLL_CFG3_0

UTMIP_PLL_CFG3_0

Offset: 0x4c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000000000000000000)

Bit	Reset	Description
26	0x0	UTMIP_PLL_REF_SRC_SEL: Source Select for UTMIPLL reference clock. 0: osc_div_clk 1: 480M output from PLLU

Bit	Reset	Description
25	REF_ENABLE	UTMIP_PLL_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
24	0x0	UTMIP_PLL_PTS: Base UTMIP_PLL test output select.
23:0	0x0	UTMIP_PLL_SETUP: Debug control bits. Default is 0. setup[6:0]: Lock Detect Controls setup[8:7]: Current source control setup [9]: Forces loop filter to VDDA/2 setup [10]: Corevdd detection override setup [11:20] TBD setup [23:21]: Phase selection on CLKOUTMUX60

5.2.176 CLK_RST_CONTROLLER_PLLREFE_BASE_0

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x00004104 (0b000000xxxxx00000010000010000100)

Bit	Reset	Description
31	DISABLE	PLLREFE_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	DISABLE	PLLREFE_ENABLE: 0 = DISABLE 1 = ENABLE
29	REF_ENABLE	PLLREFE_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
28:27	0x0	PLLREFE_KCP
26	0x0	PLLREFE_KVCO
20:16	0x0	PLLREFE_DIVP
15:8	0x41	PLLREFE_DIVN
7:0	0x4	PLLREFE_DIVM

5.2.177 CLK_RST_CONTROLLER_PLLREFE_MISC_0

Offset: 0x4c8 | Read/Write: R/W | Reset: 0xXX000000 (0bx10xx00100000000000000000000000)

Bit	R/W	Reset	Description
30	RW	0x1	PLLREFE_EN_LCKDET: 0 = DISABLE 1 = ENABLE
29	RW	0x0	PLLREFE_LOCK_OVERRIDE. Forces PLLREFE_LOCK and PLLREFE_FREQLOCK to 1.
28	RO	X	PLLREFE_FREQLOCK: 0 = frequency acquisition not achieved, 1 = frequency acquisition occurred. PLLREFE_LOCK_ENABLE must be enabled.
27	RO	X	PLLREFE_LOCK: 0 = not lock (phase+frequency), 1 = lock (phase+frequency). PLLREFE_LOCK_ENABLE must be enabled.
26:25	RW	0x0	PLLREFE_PTS: 00 = PTO is 0 (DISABLE) 01 = PTO is FO 10 = PTO is VCO out 11 = PTO is 0 0 = DISABLE 1 = FO 2 = VCO
24	RW	0x1	PLLREFE_IDDQ: 0 = The PLLREFE is powered up. 1 = Software can put the PLLREFE in IDDQ by setting this bit. 0 = OFF 1 = ON
23:0	RW	0x0	PLLREFE_SETUP

5.2.178 CLK_RST_CONTROLLER_PLLREFE_OUT_0

Offset: 0x4cc | Read/Write: R/W | Reset: 0x00020002 (0bxxxxxxxxxxxxxxx1000000000xxxxxx10)

Bit	Reset	Software Default	Description
17	ENABLE	NONE	PLLREFE_OUT2_CLKEN: PLLREFE_OUT2 branch (used in SYS chiplet as reference clock for some pLL) clk enable. 0 = DISABLE 1 = ENABLE
16	0x0	NONE	PLLREFE_OUT1_DIV_BYP: 1 = bypass PLLREFE_OUT1 divider - Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
15:8	0x0	NONE	PLLREFE_OUT1_RATIO: PLLREFE_OUT1 divider from base PLLREFE - Divide by [(N/2)+1].
1	ENABLE	DISABLE	PLLREFE_OUT1_CLKEN: PLLREFE_OUT1 divider clk enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	NONE	PLLREFE_OUT1_RSTN: PLLREFE_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.179 CLK_RST_CONTROLLER_CPU_FINETRIM_BYP_0

CPU Trimmer Registers

Bit	Name	Direction	Default Value	Description
7	byp	Read/Write	0	When asserted, the trimmer is bypassed.
6	select	Read/Write	0	When asserted, rise'rise clock propagation delay will be specified by {dr,r} fields. Otherwise, hardwired default delay will be applied. Similarly, fall'fall clock propagation delay will be specified by {df, f} fields when select is asserted.
5	dr	Read/Write	0	0: Minimal rise'rise clock propagation delay 1: Use one of the four delay steps specified by the {r} field.
4:3	r	Read/Write	0	00,01,10,11: Increment rise'rise clock delay by 1, 2, 3, or 4 steps
2	df	Read/Write	0	0: Minimal fall'fall clock propagation delay 1: Use one of the four delay steps specified by {f} field.
1:0	f	Read/Write	0	00, 01, 10, 11: Increment fall'fall clock delay by 1, 2, 3, or 4 steps

Note: The clock shaper registers are not designed to be modified "on-the-fly". These registers can only be written when the CPU clock is stopped. Failure to do so may affect the integrity of the CPU clock.

Offset: 0x4d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.2.180 CLK_RST_CONTROLLER_CPU_FINETRIM_SELECT_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	FCPU_6

Bit	Reset	Description
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.2.181 CLK_RST_CONTROLLER_CPU_FINETRIM_DR_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.2.182 CLK_RST_CONTROLLER_CPU_FINETRIM_DF_0

Offset: 0x4dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

5.2.183 CLK_RST_CONTROLLER_CPU_FINETRIM_F_0

Offset: 0x4e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:10	0x0	FCPU_6
9:8	0x0	FCPU_5
7:6	0x0	FCPU_4
5:4	0x0	FCPU_3
3:2	0x0	FCPU_2
1:0	0x0	FCPU_1

5.2.184 CLK_RST_CONTROLLER_CPU_FINETRIM_R_0

Offset: 0x4e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:10	0x0	FCPU_6
9:8	0x0	FCPU_5
7:6	0x0	FCPU_4
5:4	0x0	FCPU_3
3:2	0x0	FCPU_2

Bit	Reset	Description
1:0	0x0	FCPU_1

5.2.185 CLK_RST_CONTROLLER_PLLC2_BASE_0

Offset: 0x4e8 | Read/Write: R/W | Reset: 0x0X108002 (0b000xxxx00001xx00100000xx00000010)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC2_BYPASS: DIRECT. 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC2_ENABLE: DIRECT. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC2_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. To go into the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC2_FREQ_LOCK: 0 = not lock, 1 = frequency is locked.
26	RO	X	PLLC2_ LOCK: 0 = not locked, 1 = locked.
24:20	RW	0x1	PLLC2_DIVP: DIRECT. 0 = post divider (divide by N+1). “DIRECT”
17:10	RW	0x20	PLLC2_DIVN: PLL feedback divider LATCHED”
7:0	RW	0x2	PLLC2_DIVM: PLL input divider. DIRECT.

5.2.186 CLK_RST_CONTROLLER_PLLC2_MISC_0_0

Offset: 0x4ec | Read/Write: R/W | Reset: 0x40000000 (0bx1xxxxxxxxx00000000000000000x00)

Bit	Reset	Software Default	Description
30	0x1	NONE	PLLC2_RESET: Reset for digital logic of the PLL. DIRECT. 0 = Out of reset, 1 = Reset asserted. 0 = DISABLE 1 = ENABLE
19:4	0x0	0x8000	PLLC2_EXT_FRU
3	0x0	NONE	PLLC2_PTS: Base PLLC2 test output select. 0 = PTO is 0 1 = PTO is FO 0 = DISABLE 1 = FO
1:0	0x0	NONE	PLLC2_LOOP_CTRL

5.2.187 CLK_RST_CONTROLLER_PLLC2_MISC_1_0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1xxxxxxxxxxxxx0000xx00000000)

Bit	Reset	Description
27	0x1	PLLC2_IDDQ: 0 = OFF 1 = ON
13:10	0x0	PLLC2_EXT_SUBINT
7:0	0x0	PLLC2_DIVN_FRAC

5.2.188 CLK_RST_CONTROLLER_PLLC2_MISC_2_0

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x1f720005 (0b00011111011100100000000xxx0x101)

Bit	Reset	Software Default	Description
31:24	0x1f	NONE	PLLC2_PLL_LD_MEM
23:20	0x7	NONE	PLLC2_PLL_FRUG_HIGH
19:16	0x2	NONE	PLLC2_PLL_FRUG_LOW
15:8	0x0	0xf	PLLC2_FLL_LD_MEM
4	0x0	NONE	PLLC2_FLL_DIV
2:0	0x5	NONE	PLLC2_FLL_FRUG

5.2.189 CLK_RST_CONTROLLER_PLLC2_MISC_3_0

Offset: 0x4f8 | Read/Write: R/W | Reset: 0x000000c4 (0bxxxxxxxx00000000000000000011000100)

Bit	Reset	Description
23:8	0x0	PLLC2_SETUP
7:6	0x3	PLLC2_KP
5:4	0x0	PLLC2_LDIV
3:0	0x4	PLLC2_PLL_LD_TOL

5.2.190 CLK_RST_CONTROLLER_PLLC3_BASE_0

Offset: 0x4fc | Read/Write: R/W | Reset: 0x0X108002 (0b000xxxx00001xx00100000xx00000010)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC3_BYPASS: DIRECT. 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC3_ENABLE: DIRECT. 0 = disable, 1 = enable. Will invert and connect to IDDQ. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC3_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. To enter the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC3_FREQ_LOCK: 0 = not locked, 1 = frequency is locked.
26	RO	X	PLLC3_ LOCK: 0 = not locked, 1 = locked.
24:20	RW	0x1	PLLC3_DIVP: 0 = post divider (divide by N+1). DIRECT.
17:10	RW	0x20	PLLC3_DIVN: PLL feedback divider. LATCHED.
7:0	RW	0x2	PLLC3_DIVM: PLL input divider. DIRECT

5.2.191 CLK_RST_CONTROLLER_PLLC3_MISC_0_0

Offset: 0x500 | Read/Write: R/W | Reset: 0x40000000 (0bx1xxxxxxxxxx00000000000000000x00)

Bit	Reset	Software Default	Description
30	0x1	NONE	PLLC3_RESET: Reset for digital logic of the PLL. DIRECT. 0 = Out of reset, 1 = Reset asserted. 0 = DISABLE 1 = ENABLE
19:4	0x0	0x8000	PLLC3_EXT_FRU

Bit	Reset	Software Default	Description
3	0x0	NONE	PLLC3_PTS: Base PLLC3 test output select. 0 = PTO is 0 1 = PTO is FO 0 = DISABLE 1 = FO
1:0	0x0	NONE	PLLC3_LOOP_CTRL

5.2.192 CLK_RST_CONTROLLER_PLLC3_MISC_1_0

Offset: 0x504 | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1xxxxxxxxxxxxx0000xx00000000)

Bit	Reset	Description
27	0x1	PLLC3_IDDQ: 0 = OFF 1 = ON
13:10	0x0	PLLC3_EXT_SUBINT
7:0	0x0	PLLC3_DIVN_FRAC

5.2.193 CLK_RST_CONTROLLER_PLLC3_MISC_2_0

Offset: 0x508 | Read/Write: R/W | Reset: 0x1f720005 (0b00011111011100100000000xxx0x101)

Bit	Reset	Software Default	Description
31:24	0x1f	NONE	PLLC3_PLL_LD_MEM
23:20	0x7	NONE	PLLC3_PLL_FRUG_HIGH
19:16	0x2	NONE	PLLC3_PLL_FRUG_LOW
15:8	0x0	0xf	PLLC3_FLL_LD_MEM
4	0x0	NONE	PLLC3_FLL_DIV
2:0	0x5	NONE	PLLC3_FLL_FRUG

5.2.194 CLK_RST_CONTROLLER_PLLC3_MISC_3_0

Offset: 0x50c | Read/Write: R/W | Reset: 0x000000c4 (0bxxxxxxxx00000000000000000011000100)

Bit	Reset	Description
23:8	0x0	PLLC3_SETUP
7:6	0x3	PLLC3_KP
5:4	0x0	PLLC3_LDIV
3:0	0x4	PLLC3_PLL_LD_TOL

5.2.195 CLK_RST_CONTROLLER_PLLX_MISC_1_0

Offset: 0x510 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Software Default	Description
23:0	0x0	0x20	PLLX_SETUP: Base PLLX test output select.

5.2.196 CLK_RST_CONTROLLER_PLLX_MISC_2_0

Offset: 0x514 | Read/Write: R/W | Reset: 0x0000000X (0b00000000000000000000000xxx0xxx0)

Bit	R/W	Reset	Software Default	Description
31:24	RW	0x0	0x8	PLLX_DYNRAMP_STEPB: MISC_2 PLLX DYNRAMP

Bit	R/W	Reset	Software Default	Description
23:16	RW	0x0	0x12	PLLX_DYNRAMP_STEPA: MISC_2 PLLX DYNRAMP
15:8	RW	0x0	NONE	PLLX_NDIV_NEW: MISC_2 PLLX DYNRAMP
4	RW	0x0	NONE	PLLX_LOCK_OVERRIDE: MISC_2 PLLX DYNRAMP
3	RO	X	NONE	PLLX_PLL_FREQLOCK: MISC_2 PLLX DYNRAMP
2	RO	X	NONE	PLLX_DYNRAMP_DONE: MISC_2 PLLX DYNRAMP
0	RW	0x0	NONE	PLLX_EN_DYNRAMP: MISC_2 PLLX DYNRAMP

5.2.197 CLK_RST_CONTROLLER_PLLX_MISC_3_0

Offset: 0x518 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxx000000000xxxx0000xxxx1000)

Bit	Reset	Description
24	0x0	PLLX_PRB_OBS_SYS_SEL: Select SYS Observation clock as PLLX Probe input
23:16	0x0	PLLX_PRB_OBS_SEL: 0 = CSITE 1 = FCPU0_LEAF 2 = FCPU1_LEAF 3 = FCPU2_LEAF 4 = FCPU3_LEAF 5 = MC 6 = MSELECT 7 = FTOP_SHAPER
11:10	0x0	PLLX_PRB_DRV_CTRL
9:8	0x0	PLLX_SEL_PRB
3	0x1	PLLX_IDDQ:0: The PLLX is powered up. 1:Software can put the PLLX in IDDQ by setting this bit.
2:1	0x0	PLLX_KCP: Charge Pump Gain control
0	0x0	PLLX_KVCO:VCO gain

5.2.198 CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0

XUSB I/O PLL Power Sequencer Input Software Control Program Guide

The software control input can be used for test coverage or driving power sequencer SM via software.

The XUSB power sequencer is driven by a primary reset input signal: reset. Reset is expected to be held static until the power-up or power-down sequence is complete.

Offset: 0x51c | Read/Write: R/W | Reset: 0x0X0X380d (0b0000xx10xxxxxxxx00111000x0001101)

Bit	R/W	Reset	Software Default	Description
31:28	RW	0x0	NONE	XUSBIO_PADPLL_IDDQ2_PADPLL_RCAL_DLY: PT1: Delay from XUSBIO PAD PLL IDDQ DE assertion to the PAD PLL RCAL EN assertion. Should be at least 50ns and FSM delays take care of it.
27:26	RO	X	NONE	XUSBIO_SEQ_STATE: XUSBIO power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	NONE	XUSBIO_SEQ_START_STATE: XUSBIO power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	ENABLE	XUSBIO_SEQ_ENABLE: XUSBIO power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
16	RO	X	NONE	XUSBIO_PADPLL_CAL_VALID:Indicator that Brick PLL calibration was done and is valid. CAL_VALID_RST bit should make this '0'.

Bit	R/W	Reset	Software Default	Description
15	RW	0x0	NONE	XUSBIO_PADPLL_REQ_PLL_RCAL_BYPASS: 0= Don't bypass Resistor calibration, 1=BYPASS Resistor calibration
14	RW	0x0	NONE	XUSBIO_PADPLL_REQ_PLL_RCAL: Trigger signal needs a rising edge on this bit to Force PLL Resistor calibration
13	RW	0x1	NONE	XUSBIO_PADPLL_SLEEP_IDDQ: IDDQ value during SLEEP mode
12:11	RW	0x3	NONE	XUSBIO_PADPLL_SLEEP_VAL: select type of PLL sleep states from 1 to 3
10	RW	0x0	NONE	XUSBIO_PADPLL_CAL_VALID_RST: 0= Do not reset, 1=reset the CAL VALID
9	RW	0x0	NONE	XUSBIO_PADPLL_REQ_PLL_CAL_RESET: its a trigger signal need a rising edge on this bit to reset PLL VCO calibration
8	RW	0x0	NONE	XUSBIO_PADPLL_REQ_PLL_CAL: its a trigger signal need a rising edge on this bit to Force PLL VCO calibration
6	RW	0x0	0x1	XUSBIO_PADPLL_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLL
5	RW	0x0	NONE	XUSBIO_SEQ_RESET_INPUT_VALUE: 0=XUSBIO PLL ON, 1=XUSB IOPLL OFF. Software controls the state machine by setting this and the XUSBIO_SEQ_IN_SWCTL bit
4	RW	0x0	NONE	XUSBIO_SEQ_IN_SWCTL: 0=seq input by hardware, 1=seq input by software.
3	RW	ENABLE	NONE	XUSBIO_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when XUSBIO_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	0x0	XUSBIO_CLK_ENABLE_SWCTL: 0=ioclks enabled by hardware, 1=ioclks enabled by software
1	RW	0x0	NONE	XUSBIO_PADPLL_RESET_OVERRIDE_VALUE: 0=XUSBIO PLL on, 1=PLL reset. Override value used only when XUSBIO_PADPLL_RESET_SWCTL is set
0	RW	0x1	0x0	XUSBIO_PADPLL_RESET_SWCTL: 0=Reset by hardware, 1=Reset by software

5.2.199 CLK_RST_CONTROLLER_XUSBIO_PLL_CFG1_0

The range value is the recommended range. All counters support 0-255 µs range in 1 µs steps.

Offset: 0x520 │ Read/Write: R/W │ Reset: 0x00000014 (0bxxxxxxxx00000000000000000010100)

Bit	Reset	Description
23:16	0x0	XUSBIO_PADPLL_RESET2_PADPLL_IDDQ_DLY: PT1: Delay from XUSBIO PAD PLL SLEEP assertion to the PAD PLL IDDQ assertion. Value is greater than 50 ns, which the design takes care of.
15:8	0x0	XUSBIO_PADPLL_IDDQ2_PADPLL_RESET_DLY: PT2: Delay from XUSBIO PAD PLL SLEEP de-assertion (or CAL_DONE) to the PAD PLL enable de-assertion. Range is 0 - 10 µs.
7:0	0x14	XUSBIO_PADPLL_PU_POST_DLY: PT3: Delay from XUSBIO PAD PLL Enable assertion to the PLL LOCK. Range is 1 -20 µs.

5.2.200 CLK_RST_CONTROLLER_PLLE_AUX1_0

The range value is the recommended range. All counters support 0-255 step range.

Offset: 0x524 │ Read/Write: R/W │ Reset: 0x00000105 (0bxxxxxxxxxxxxxxxx0000000100000101)

Bit	Reset	Description
15:8	0x1	PLLE_INTRESET_DLY: Delay between PLLE SSC_BYP -> PLLE INTERP_RESET [delay 300 ns-500 ns minimum]. 300 ns to 500 ns delay is required for the interpolator biasing to stabilize. The state machine can give a minimum 1 µs of delay.
7:0	0x5	PLLE_ENABLE_DLY: Delay from PLLE IDDQ de-assertion to PLLE ENABLE assertion. [~5 µs] IDDQ enables Vregulator. ENABLE starts PLL. It takes 5 µs for the voltage regulator to come up. The voltage regulator needs to be up before the PLL is started.

5.2.201 CLK_RST_CONTROLLER_PLLP_RESHIFT_0

Offset: 0x528 | Read/Write: R/W | Reset: 0x00000053 (0bxxxxxxxxxxxxxxxxxxxxxxx0001010011)

Bit	Reset	Description
9:2	0x14	PLLP_OUT0_RATIO: PLLP_OUT0 divider from base PLLP. Divide by [(N/2)+1]. The default is 408/8=51 MHz.
1	ENABLE	PLLP_OUT0_CLKEN: PLLP_OUT0 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLP_OUT0_RSTN: PLLP_OUT0 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.202 CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0

Offset: 0x52c | Read/Write: R/W | Reset: 0xXX00000f (0bxxxxxx10xxxxxxxxxxxxxxxx00001111)

Bit	R/W	Reset	Software Default	Description
31	RO	X	NONE	UTMIPLL_LOCK: 0 = not lock, 1 = lock. (Phase and Frequency)
27:26	RO	X	NONE	UTMIPLL_SEQ_STATE: UTMIPLL power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	NONE	UTMIPLL_SEQ_START_STATE: UTMIPLL power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	ENABLE	UTMIPLL_SEQ_ENABLE: UTMIPLL power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x0	NONE	UTMIPLL_IDDQ_PD_INCLUDE: While powering down through the hardware state machine, puts the PLL in IDDQ as well to save more power.
6	RW	0x0	0x1	UTMIPLL_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLL and programmable delay on top of that.
5	RW	0x0	NONE	UTMIPLL_SEQ_RESET_INPUT_VALUE: 0=UTMIPLL ON, 1=UTMIPLL OFF. Software controls the state machine by setting this and UTMIPLL_SEQ_IN_SWCTL bit
4	RW	0x0	NONE	UTMIPLL_SEQ_IN_SWCTL: 0=seq input by hardware, 1=seq input by software
3	RW	ENABLE	NONE	UTMIPLL_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when UTMIPLL_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	0x0	UTMIPLL_CLK_ENABLE_SWCTL: 0=UTMIP clocks enable by hardware, 1=UTMIP clocks enable by software
1	RW	0x1	0x0	UTMIPLL_IDDQ_OVERRIDE_VALUE: 0=PLL not in IDDQ mode, 1=PLL in IDDQ mode. Override value used only when UTMIPLL_IDDQ_SWCTL is set
0	RW	0x1	NONE	UTMIPLL_IDDQ_SWCTL: 0=IDDQ by hardware, 1=IDDQ by software.

5.2.203 CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0

Offset: 0x530 | Read/Write: R/W | Reset: 0x0X00008d (0bxxx0xx10xxxxxxxxxxxxxxxx100011x1)

Bit	R/W	Reset	Software Default	Description
28	RW	0x0	0x0	PLLU_IDDQ_PD_INCLUDE: While powering down through hardware state machine, puts the PLLU in IDDQ as well to save more power.

Bit	R/W	Reset	Software Default	Description
27:26	RO	X	NONE	PLLU_SEQ_STATE: PLLU power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	NONE	PLLU_SEQ_START_STATE: PLLU power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	0x0	PLLU_SEQ_ENABLE: PLLU power sequencer enable. Start state is loaded on the first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x1	NONE	PLLU_USE_SWITCH_DETECT: 0=Use programmable delay, 1=Use hardware switch detect logic.
6	RW	0x0	0x0	PLLU_USE_LOCKDET: 0=Use programmable delay, 1=Use lockdet signals from PLL and programmable delay on top of that.
5	RW	0x0	NONE	PLLU_SEQ_RESET_INPUT_VALUE: 0=PLLU ON, 1=PLLU OFF. Software controls the state machine by setting this and the PLLU_SEQ_IN_SWCTL bit
4	RW	0x0	NONE	PLLU_SEQ_IN_SWCTL: 0=Seq input by hardware, 1=Seq input by software
3	RW	ENABLE	NONE	PLLU_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when PLLU_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	0x0	PLLU_CLK_ENABLE_SWCTL: 0=PLLU clocks enabled by hardware, 1=PLLU clocks enabled by software
0	RW	0x1	0x0	PLLU_CLK_SWITCH_SWCTL: 0=Control SS/FS clock frequency through hardware, 1=Control FS/SS clock frequency through software

5.2.204 CLK_RST_CONTROLLER_XUSB_PLL_CFG0_0

Note that default value is the recommended range.

Offset: 0x534 │ Read/Write: R/W │ Reset: 0x8002a80f (0b10000000000000101010100000001111)

Bit	Reset	Software Default	Description
31:24	0x80	NONE	PLLU_CLK_SWITCH_DLY: Delay from SS Clock source change to the actual frequency change to 32 kHz clock. Range is ~100 µs.
23:14	0xa	0x0	PLLU_LOCK_DLY: Delay from PLLU ENABLE assertion to the PLL LOCK. Range is 7 µs-10 µs.
13:10	0xa	NONE	UTMIPLL_IDDQ2_ENABLE_DLY: Delay from UTMIPLL IDDQ de-assertion to the UTMIPLL ENABLE assertion. Range ~5 µs.
9:0	0xf	0x0	UTMIPLL_LOCK_DLY: Delay from UTMIPLL ENABLE assertion to the PLL LOCK. Range 10-15 µs.

5.2.205 CLK_RST_CONTROLLER_CLK_CPU_MISC_0

Offset: 0x53c │ Read/Write: R/W │ Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CPU_ATCLK_RATIO: Clock divider ratio for the ATB clocks in CCPLEX 000 = div-by-2. 001 = div-by-3. 010 = div-by-4. 011 = div-by-5. 100 = div-by-6. 101 = div-by-7. 110 = div-by-8. Other values = Reserved

5.2.206 CLK_RST_CONTROLLER_CLK_CPUG_MISC_0

Offset: 0x540 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CPUG_ATCLK_RATIO: Clock divider ratio for the ATB clocks in CCPLEX 000 = div-by-2. 001 = div-by-3. 010 = div-by-4. 011 = div-by-5. 100 = div-by-6. 101 = div-by-7. 110 = div-by-8. Other values = Reserved

5.2.207 CLK_RST_CONTROLLER_PLLX_HW_CTRL_CFG_0

Offset: 0x548 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000001)

Bit	Reset	Description
6	DISABLE	FORCE_FSM_CLEAR: 0 = DISABLE 1 = ENABLE
5:4	REF_DIVM	SLOWDOWN_RESERVED: 0 = REF_DIVM 1 = REF_DIVM_DIV4 2 = SCLK_DIV1 3 = SCLK_DIV16
3	SLOW	RAMP_MODE: 0 = SLOW 1 = FAST
2	DISABLE	SEQ_TRIGGER: 0 = DISABLE 1 = ENABLE
1	DISABLE	SW_OVERRIDE: 0 = DISABLE 1 = ENABLE
0	ENABLE	SWCTL: 0 = DISABLE 1 = ENABLE

5.2.208 CLK_RST_CONTROLLER_PLLX_SW_RAMP_CFG_0

Offset: 0x54c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DYNRAMP_STEPA: PLL ramp rate
23:16	0x0	DYNRAMP_STEPB: PLL ramp rate

Bit	Reset	Description
15:8	0x0	NDIV_NEW: PLL ramp rate
7:0	0x0	NDIV: PLL ramp rate

5.2.209 CLK_RST_CONTROLLER_PLLX_HW_CTRL_STATUS_0

Offset: 0x550 | Read/Write: RO | Reset: 0xXX000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	FSM_STATE: 0 = DISABLE 1 = SEQ_AWAIT 2 = SEQ_FAST_BEGIN 3 = SEQ_FAST_BUSY0 4 = SEQ_FAST_BUSY1 5 = SEQ_FAST_BUSY2 6 = SEQ_FAST_BUSY3 7 = SEQ_SLOW_BEGIN 8 = SEQ_SLOW_BUSY0 9 = SEQ_SLOW_BUSY1 10 = SEQ_SLOW_BUSY2 11 = SEQ_SLOW_BUSY3 12 = SEQ_DONE 13 = ILLEGAL
27:26	X	CFG_REG_SELECT: 0 = NONE 1 = Hardware 2 = Software
10	X	HW_RAMP_DONE: 0 = FALSE 1 = TRUE
9	X	SW_RAMP_DONE: 0 = FALSE 1 = TRUE
8	X	LONG_LATENCY_THROTTLE: 0 = DISABLE 1 = ENABLE
7	X	HW_RESTORE_EN: 0 = DISABLE 1 = ENABLE
6	X	HW_THROTTLE_EN: 0 = DISABLE 1 = ENABLE
5	X	SW_RAMP_STATUS: 0 = DONE 1 = INPROGRESS
4	X	HW_RAMP_STATUS: 0 = DONE 1 = INPROGRESS
3:2	X	SEQ_STATUS: 0 = SEQ_DIABLE 1 = SEQ_ENABLE 2 = SEQ_BUSY_FAST_MODE 3 = SEQ_BUSY_SLOW_MODE
1	X	SW_OVERRIDE_STATUS: 0 = DISABLE 1 = ENABLE
0	X	SWCTL_STATUS: 0 = DISABLE 1 = ENABLE

5.2.210 CLK_RST_CONTROLLER_SPARE_REG0_0

Offset: 0x55c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Software Default	Description
31:6	0x0	NONE	VAL
5	0x0	NONE	DIVIDER_FSM_RESERVED: Disable ability to change rate and ratio while module clock is off. Module should be off when changing this value.
4	0x0	NONE	VAL1
3:2	0x0	0x1	CLK_M_DIVISOR: N = Divide by (n+1).
1	0x0	NONE	TMR_CLKEN_STICKY
0	0x0	NONE	EMC_LATENCY_OVERRIDE

5.2.211 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_DMIC1_0

Audio sync clock source select

Offset: 0x560 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK.
3:0	0x0	SYNC_CLK_RATE: 0000 = reserved 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = reserved 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.212 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_DMIC2_0

Audio sync clock source select

Offset: 0x564 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK. Also
3:0	0x0	SYNC_CLK_RATE: 0000 = Reserved 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = reserved 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.213 CLK_RST_CONTROLLER_PLLD2_SS_CFG_0

Offset: 0x570 | Read/Write: R/W | Reset: 0x1XX00000 (0b00010xxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD2_EN_SDM:Fractional N divider enable 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD2_EN_SSC: Spread spectrum enable 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
29	RW	0x0	PLLD2_EN_DITHER2
28	RW	0x1	PLLD2_EN_DITHER
27	RW	0x0	PLLD2_SDM_RESET
25:23	RO	X	PLLD2_SDM_TEST_OUT

5.2.214 CLK_RST_CONTROLLER_PLLD2_SS_CTRL1_0

Offset: 0x574 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLD2_SDM_SSC_MAX
15:0	0x0	PLLD2_SDM_SSC_MIN

5.2.215 CLK_RST_CONTROLLER_PLLD2_SS_CTRL2_0

Offset: 0x578 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PLLD2_SDM_SSC_STEP
15:0	0x0	PLLD2_SDM_DIN

5.2.216 CLK_RST_CONTROLLER_PLLDP_BASE_0

PLLDP Registers

vcoclock Frequency:

- Normal mode: EN_SDM=DISABLE, SDM_RESET=0

  Fvco = Fref / MDIV * NDIV

- Fractional NDIV mode: EN_SDM=ENABLE, EN_SSC=DISABLE, SDM_RESET=0

  Fvco = Fref / MDIV * (NDIV + 0.5 + SDM_DIN / 8192)

- Spread Spectrum mode: EN_SDM=ENABLE, EN_SSC=ENABLE, SDM_RESET=0

  Fvco_max = Fref / MDIV * (NDIV + 0.5 + SDM_SSC_MAX / 8192), SDM_SSC_MAX < 0x300

  Fvco_min = Fref / MDIV * (NDIV + 0.5 + SDM_SSC_MIN / 8192), SDM_SSC_MIN > -0x300 (0xD000)

  Fvco_step = 2 * (Fmod * MDIV / Fref) * (SDM_SSC_MAX-SDM_SSC_MIN), Fmod-modulation frequency,e.g., 33KHz

  = SDM_SSC_STEP[11:0] / (SDM_SSC_STEP[15:12] + 1)

Offset: 0x590 | Read/Write: R/W | Reset: 0xXX041401 (0b000xx000000001x00001010000000001)

Bit	R/W	Reset	Software Default	Description
31	RW	DISABLE	NONE	PLLDP_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	NONE	PLLDP_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ ENABLE	NONE	PLLDP_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
28	RO	X	NONE	PLLDP_FREQLOCK: 0 = not lock, 1 = lock frequency
27	RO	X	NONE	PLLDP_LOCK: 0 = not lock, 1 = lock frequency + phase

Bit	R/W	Reset	Software Default	Description
26:25	RW	0x0	NONE	PLLDP_REF_SRC_SEL:Reference source select sel[0]=1 => ref_src = PLLREFE_CLKOUTsel[0]=0 && sel[1]=1 => ref_src = pllP_out0sel[0]=0 && sel[1]=0 => ref_src = osc_div_clk
24	RW	0x0	NONE	PLLDP_LOCK_OVERRIDE:Forces PLL_LOCK to 1
23:19	RW	0x0	0x3	PLLDP_PDIV: PL divider
18	RW	0x1	NONE	PLLDP_IDDQ: 0 = The PLL is powered up, 1 = Software can put the PLL in IDDQ by setting this bit 0 = OFF 1 = ON
16	RW	0x0	NONE	PLLDP_PTS: Base PLLDP test output select. 0 = PTO is 0 1 = PTO is FO
15:8	RW	0x14	0x1c	PLLDP_NDIV:N divider
7:0	RW	0x1	NONE	PLLDP_MDIV:M divider

5.2.217 CLK_RST_CONTROLLER_PLLDP_MISC_0

Offset: 0x594 | Read/Write: R/W | Reset: 0x40000000 (0bx1xxx00000000000000000000000000)

Bit	R/W	Reset	Software Default	Description
30	RW	ENABLE	NONE	PLLDP_EN_LCKDET: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
26:25	RW	0x0	NONE	PLLDP_KCP: Charge pump gain control
24	RW	0x0	NONE	PLLDP_KVCO:VCO gain
23:0	RW	0x0	0x20	PLLDP_SETUP:setup[23:0]

5.2.218 CLK_RST_CONTROLLER_PLLDP_SS_CFG_0

Offset: 0x598 | Read/Write: R/W | Reset: 0x1XX00000 (0b00010xxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Software Default	Description
31	RW	DISABLE	NONE	PLLDP_EN_SDM: Fractional N divider enable 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	NONE	PLLDP_EN_SSC: Spread spectrum enable 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	0x0	NONE	PLLDP_EN_DITHER2
28	RW	0x1	0x0	PLLDP_EN_DITHER
27	RW	0x0	NONE	PLLDP_SDM_RESET
25:23	RO	X	NONE	PLLDP_SDM_TEST_OUT

5.2.219 CLK_RST_CONTROLLER_PLLDP_SS_CTRL1_0

Offset: 0x59c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Software Default	Description
31:16	0x0	0xf400	PLLDP_SDM_SSC_MAX
15:0	0x0	0xf0da	PLLDP_SDM_SSC_MIN

5.2.220 CLK_RST_CONTROLLER_PLLDP_SS_CTRL2_0

Offset: 0x5a0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Software Default	Description
31:16	0x0	0x2004	PLLDP_SDM_SSC_STEP
15:0	0x0	0x2004	PLLDP_SDM_DIN

5.2.221 CLK_RST_CONTROLLER_PLLC4_BASE_0

Offset: 0x5a4 | Read/Write: R/W | Reset: 0xXX0c2302 (0b000xx00000001100001000110000010)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC4_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC4_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC4_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
28	RO	X	PLLC4_FREQLOCK: 0 = not lock, 1 = lock feq
27	RO	X	PLLC4_LOCK: 0 = not lock, 1 = lock feq + phase
26:25	RW	0x0	PLLC4_REF_SRC_SEL:Reference source select sel[0]=1 => ref_src = PLLREFE_CLKOUTsel[0]=0 && sel[1]=1 => ref_src = pllP_out0sel[0]=0 && sel[1]=0 => ref_src = osc_div_clk
24	RW	0x0	PLLC4_LOCK_OVERRIDE:Forces PLL_LOCK to 1
23:19	RW	0x0	PLLC4_DIVP:PL divider
18	RW	0x1	PLLC4_IDDQ: 0 The PLL is powered up, : 0oftware can put the PLL in IDDQ by setting this bit 0 = OFF 1 = ON
17:16	RW	0x0	PLLC4_PTS: Base PLLC4 test output select. 00 = PTO is 0. 01 = PTO is FO. 10 = PTO is FO_DIV3. 11 = PTO is FO_DIV5 0 = DISABLE 1 = FO 2 = FO_DIV3 3 = FO_DIV5
15:8	RW	0x23	PLLC4_DIVN:N divider
7:0	RW	0x2	PLLC4_DIVM:M divider

5.2.222 CLK_RST_CONTROLLER_PLLC4_MISC_0

Offset: 0x5a8 | Read/Write: R/W | Reset: 0x40000000 (0bx1xxx00000000000000000000000000)

Bit	R/W	Reset	Description
30	RW	0x1	PLLC4_EN_LCKDET: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
26:25	RW	0x0	PLLC4_KCP:Charge pump gain control
24	RW	0x0	PLLC4_KVCO:VCO gain
23:0	RW	0x0	PLLC4_SETUP:setup[23:0]

5.2.223 CLK_RST_CONTROLLER_CLK_SPARE0_0

Offset: 0x5c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

5.2.224 CLK_RST_CONTROLLER_CLK_SPARE1_0

Offset: 0x5c8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

5.2.225 CLK_RST_CONTROLLER_GPU_ISOB_CTRL_0

Offset: 0x5cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	CAR_ISOB_EN: Enable/Disable the idle slowdown on boot (ISoB) to the GPU 0 = DISABLE 1 = ENABLE

5.2.226 CLK_RST_CONTROLLER_PLLC_MISC_2_0

Offset: 0x5d0 | Read/Write: R/W | Reset: 0x1f720005 (0b00011111011100100000000xxx0x101)

Bit	Reset	Software Default	Description
31:24	0x1f	NONE	PLLC_PLL_LD_MEM
23:20	0x7	NONE	PLLC_PLL_FRUG_HIGH
19:16	0x2	NONE	PLLC_PLL_FRUG_LOW
15:8	0x0	0xf	PLLC_FLL_LD_MEM
4	0x0	NONE	PLLC_FLL_DIV
2:0	0x5	NONE	PLLC_FLL_FRUG

5.2.227 CLK_RST_CONTROLLER_PLLC_MISC_3_0

Offset: 0x5d4 | Read/Write: R/W | Reset: 0x000000c4 (0bxxxxxxxx00000000000000011000100)

Bit	Reset	Description
23:8	0x0	PLLC_SETUP
7:6	0x3	PLLC_KP
5:4	0x0	PLLC_LDIV
3:0	0x4	PLLC_PLL_LD_TOL

5.2.228 CLK_RST_CONTROLLER_PLLA_MISC2_0

Offset: 0x5d8 | Read/Write: R/W | Reset: 0x0X000000 (0bxxxxx00x00000000000000000000000)

Bit	R/W	Reset	Description
26	RW	0x0	PLLA_EN_SDM
25	RW	0x0	PLLA_EN_DYNRAMP
24	RO	X	PLLA_DYNRAMP_DONE
23:16	RW	0x0	PLLA_DYNRAMP_STEPB
15:8	RW	0x0	PLLA_DYNRAMP_STEPA
7:0	RW	0x0	PLLA_NDIV_NEW

5.2.229 CLK_RST_CONTROLLER_PLLC4_OUT_0

Offset: 0x5e4 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxx000000000xxxxxx10)

Bit	Reset	Description
16	0x0	PLLC4_OUT3_DIV_BYP: 1 = bypass PLLC4_OUT3 divider - Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
15:8	0x0	PLLC4_OUT3_RATIO: PLLC4_OUT3 divider from base PLLC4 - Divide by [(N/2)+1].
1	ENABLE	PLLC4_OUT3_CLKEN: PLLC4_OUT3 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
0	RESET_ENABLE	PLLC4_OUT3_RSTN: PLLC4_OUT3 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.230 CLK_RST_CONTROLLER_PLLMB_BASE_0

Offset: 0x5e8 | Read/Write: R/W | Reset: 0x0X002a02 (0bx0xxxxx00000xxxx0010101000000010)

Bit	R/W	Reset	Description
30	RW	DISABLE	PLLMB_ENABLE: 0 = disable, 1 = enable. (will invert and connect to IDDQ) 0 = DISABLE 1 = ENABLE
27	RO	X	PLLMB_LOCK: 0 = not lock, 1 = lock. (Phase and Frequency)
26	RO	X	PLLMB_FREQ_LOCK: 0 = not lock, 1 = lock. Frequency lock only
24:20	RW	0x0	PLLMB_DIVP: PLL post divider
15:8	RW	0x2a	PLLMB_DIVN: PLL feedback divider.
7:0	RW	0x2	PLLMB_DIVM: PLL input divider.

5.2.231 CLK_RST_CONTROLLER_PLLMB_MISC1_0

Offset: 0x5ec | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxxxxxx0010000000000000000)

Bit	Reset	Description
18	0x0	PLLMB_LOCK_OVERRIDE:forces lock to 1
17	0x0	PLLMB_IDDQ
16	0x1	PLLMB_EN_LCKDET: 0 = DISABLE 1 = ENABLE
15:0	0x0	PLLMB_SETUP:SETUP fields

5.2.232 CLK_RST_CONTROLLER_PLLX_MISC_4_0

Offset: 0x5f0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	PLLX_EN_SDM: Enable Sigma Delta Modulator feature.
15:0	0x0	PLLX_SDM_DIN: Fractional input to Sigma Delta Modulator. Range 0xF000 - 0x1000.

5.2.233 CLK_RST_CONTROLLER_PLLX_MISC_5_0

Offset: 0x5f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	PLLX_SDM_DIN_NEW: Fractional input new to Sigma Delta Modulator. Range 0xF000 - 0x1000.

5.2.234 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_HOST_0

Offset: 0x600 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_CORE_HOST_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 5 = PLLREFE_CLKOUT
7:0	0x0	XUSB_CORE_HOST_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.235 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FALCON_0

Offset: 0x604 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_FALCON_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 5 = PLLREFE_CLKOUT
7:0	0x0	XUSB_FALCON_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.236 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FS_0

Offset: 0x608 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Software Default	Description
31:29	CLK_M	FO_48M	XUSB_FS_CLK_SRC: 000 = clk_m, 010 = FO_48M, 100 = pllP_out0, 110 = HSIC_480 0 = CLK_M 2 = FO_48M 4 = PLLP_OUT0 6 = HSIC_480
7:0	0x0	NONE	XUSB_FS_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.237 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_DEV_0

Offset: 0x60c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_CORE_DEV_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 5 = PLLREFE_CLKOUT
7:0	0x0	XUSB_CORE_DEV_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.238 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_SS_0

Offset: 0x610 | Read/Write: R/W | Reset: 0x04000000 (0b000xx100xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Software Default	Description
31:29	CLK_M	HSIC_480	XUSB_SS_CLK_SRC: 000 = clk_m, 001 = pllRefe_OUT, 010 = clk_s, 011 = HSIC_480 0 = CLK_M 1 = PLLREFE_CLKOUT 2 = CLK_S 3 = HSIC_480
26	0x1	NONE	XUSB_HS_HSICP_CLK_SEL: 0 = HS_HSICP clock is 60MHz (same as xusb_hs_clk) 1 = HS_HSICP clock is 120MHz (sourced from xusb_ss_clk switch/ divider)

Bit	Reset	Software Default	Description
25	0x0	NONE	XUSB_HS_CLK_BYPASS_SWITCH: 0 = HS clock is switch/dividers output. 1 = HS Clock is derived from PLLU 60M output directly.
24	0x0	NONE	XUSB_SS_CLK_BYPASS_SWITCH: 0 = SS clock is switch/dividers output. 1 = SS Clock is derived from osc_div clock directly.
7:0	0x0	0x6	XUSB_SS_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.239 CLK_RST_CONTROLLER_CLK_SOURCE_CILAB_0

Offset: 0x614 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	CILAB_CLK_SRC: 000 = pllP_out0, 010 = pllC_out0, 011 = pllC4_out0, 100 = pllC4_out1, 101 = pllC4_out2, 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = PLLC4_OUT1 5 = PLLC4_OUT2 6 = CLK_M
7:0	0x0	CILAB_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.240 CLK_RST_CONTROLLER_CLK_SOURCE_CILCD_0

Offset: 0x618 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	CILCD_CLK_SRC: 000 = pllP_out0. 010 = pllC_out0. 011 = pllC4_out0. 100 = pllC4_out1. 101 = pllC4_out2. 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = PLLC4_OUT1 5 = PLLC4_OUT2 6 = CLK_M
7:0	0x0	CILCD_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.241 CLK_RST_CONTROLLER_CLK_SOURCE_CILEF_0

Offset: 0x61c | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	CILEF_CLK_SRC: 000 = pllP_out0 010 = pllC_out0 011 = pllC4_out0 100 = pllC4_out1 101 = pllC4_out2 100 = clk_m 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = PLLC4_OUT1 5 = PLLC4_OUT2 6 = CLK_M
7:0	0x0	CILEF_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.242 CLK_RST_CONTROLLER_CLK_SOURCE_DSIA_LP_0

Offset: 0x620 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DSIA_LP_CLK_SRC: 000 = pllP_out0. 010 = pllC_out0. 011 = pllC4_out0. 100 = pllC4_out1. 101 = pllC4_out2. 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = PLLC4_OUT1 5 = PLLC4_OUT2 6 = CLK_M
7:0	0x0	DSIA_LP_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.243 CLK_RST_CONTROLLER_CLK_SOURCE_DSIB_LP_0

Offset: 0x624 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DSIB_LP_CLK_SRC: 000 = pllP_out0. 010 = pllC_out0. 011 = pllC4_out0. 100 = pllC4_out1. 101 = pllC4_out2. 110 = clk_m 0 = PLLP_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 4 = PLLC4_OUT1 5 = PLLC4_OUT2 6 = CLK_M
7:0	0x0	DSIB_LP_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.244 CLK_RST_CONTROLLER_CLK_SOURCE_ENTROPY_0

Offset: 0x628 | Read/Write: R/W | Reset: 0x40000000 (0b010xxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31:29	CLK_M	ENTROPY_CLK_SRC: 000 = pllP_out0. 010 = clk_m. 100 = clk_s. 110 = pllE_out0 0 = PLLP_OUT0 2 = CLK_M 4 = CLK_S 6 = PLLE_OUT0
8	0x0	ENTROPY_CLK_LOCK: 1 = Lock entire CLK_SOURCE_ENTROPY register, can only be unlocked by reset.
7:0	0x0	ENTROPY_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.245 CLK_RST_CONTROLLER_CLK_SOURCE_DVFS_REF_0

Offset: 0x62c | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DVFS_REF_CLK_SRC: 000 = pllP_out0 001 = pllC2_out0 010 = pllC_out0 011 = pllC4_out0 101 = pllC4_out1 110 = clk_m 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
28	0x1	DVFS_REF_MASTER_CLKEN: Reserved.

Bit	Reset	Description
7:0	0x0	DVFS_REF_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.246 CLK_RST_CONTROLLER_CLK_SOURCE_DVFS_SOC_0

Offset: 0x630 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DVFS_SOC_CLK_SRC: 000 = pllP_out0. 001 = pllC2_out0. 010 = pllC_out0. 011 = pllC4_out0. 101 = pllC4_out1. 110 = clk_m. 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
28	0x1	DVFS_SOC_MASTER_CLKEN: Reserved.
7:0	0x0	DVFS_SOC_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.247 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_LATENCY_0

Offset: 0x640 | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_LATENCY_CLK_SRC: 001 = pllC_out0. 010 = pllP_out0. 011 = clk_m. 101 = pllC4_out0. 110 = pllC4_out1. 111 = pllC4_out2 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 5 = PLLC4_OUT0 6 = PLLC4_OUT1 7 = PLLC4_OUT2
7:0	0x0	EMC_LATENCY_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.248 CLK_RST_CONTROLLER_CLK_SOURCE_SOC_THERM_0

Offset: 0x644 | Read/Write: R/W | Reset: 0x40000000 (0b010xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	SOC_THERM_CLK_SRC: 000 = clk_m. 001 = pllC_out0. 010 = pllP_out0. 011 = pllA_out0. 100 = pllC2_out0. 101 = pllC4_out0. 101 = pllC4_out1. 101 = pllC4_out2 0 = CLK_M 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC4_OUT0 6 = PLLC4_OUT1 7 = PLLC4_OUT2
7:0	0x0	SOC_THERM_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.249 CLK_RST_CONTROLLER_CLK_SOURCE_DMIC1_0

Offset: 0x64c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLA_OUT0	DMIC1_CLK_SRC: 000 = pllA_out0. 001 = audio SYNC_CLK 1x or 2x. 010 = pllP_out0. 011 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK 2 = PLLP_OUT0 3 = CLK_M
7:0	0x0	DMIC1_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.250 CLK_RST_CONTROLLER_CLK_SOURCE_DMIC2_0

Offset: 0x650 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLA_OUT0	DMIC2_CLK_SRC: 000 = pllA_out0. 001 = audio SYNC_CLK 1x or 2x. 010 = pllP_out0. 011 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK 2 = PLLP_OUT0 3 = CLK_M
7:0	0x0	DMIC2_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.251 CLK_RST_CONTROLLER_CLK_SOURCE_VI_SENSOR2_0

Offset: 0x658 | Read/Write: R/W | Reset: 0x80000000 (0b100xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	VI_SENSOR2_CLK_SRC: 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	VI_SENSOR2_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.252 CLK_RST_CONTROLLER_CLK_SOURCE_I2C6_0

Offset: 0x65c | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C6_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC4_out0, 101 = pllC4_out1, 110 = clk_m, 111 = pllC4_out2 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT0 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT2
15:0	0x0	I2C6_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.2.253 CLK_RST_CONTROLLER_CLK_SOURCE_MIPIBIF_0

Offset: 0x660 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	MIPIBIF_CLK_SRC: Reserved

Bit	Reset	Description
7:0	0x0	MIPIBIF_CLK_DIVISOR: Reserved

5.2.254 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_DLL_0

Offset: 0x664 | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxxxxxxx00xx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_DLL_CLK_SRC: 000 = pllM_out0 001 = pllC_out0 010 = pllP_out0 011 = clk_m 100 = pllM_out_for_emc, same as pllM_out0, no low jitter path for this. 101 = pllMb_out_for_emc, same as pllMb_out0, no low jitter path for this. 110 = pllMb_out0 111 = PLLP_UD, same as pllP_out0, no low jitter path for this. 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLMB_UD 6 = PLLMB_OUT0 7 = PLLP_UD
11:10	0x0	DDLL_CLK_SEL: DLL Selects one of the 4 inputs listed in ENUM 1 = The output of this switch is the source for DLL clock. 0 = PLLM_VCOA 1 = PLLM_VCOB 2 = EMC_DLL_SWITCH_OUT
7:0	0x0	EMC_DLL_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.255 CLK_RST_CONTROLLER_CLK_SOURCE_UART_FST_MIPI_CAL_0

Offset: 0x66c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT3	CLK72MHZ_CLK_SRC: 000 = pllP_out3 001 = pllP_out0 010 = pllC_out0 100 = pllC2_out0 110 = CLK_M 0 = PLLP_OUT3 1 = PLLC_OUT0 2 = PLLC2_OUT0 4 = PLLC2_OUT0 6 = CLK_M
7:0	0x0	UART_FST_MIPI_CAL_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.256 CLK_RST_CONTROLLER_CLK_SOURCE_VIC_0

Offset: 0x678 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	VIC_CLK_SRC: 001 = pllC_out0, 010 = pllP_out0, 011 = pllA1_out0, 100 = pllC2_out0, 101 = pllC3_out0, 110 = clk_m 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA1_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M
15:8	0x0	VIC_IDLE_DIVISOR: Divide by [(N/2)+1] if all 0's, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of VIC_CLK_DIVISOR.
7:0	0x0	VIC_CLK_DIVISOR: N = Divide by [(N/2)+1]

5.2.257 CLK_RST_CONTROLLER_PLLP_OUTC_0

Offset: 0x67c | Read/Write: R/W | Reset: 0x00030000 (0b00000000xxxxx011xxxxxxxxxxxxxxxx)

Bit	Reset	Software Default	Description
31:24	0x0	0x2	PLLP_OUT5_RATIO: PLLP_OUT5 divider from base PLLP. Divide by [(N/2)+1].
18	DISABLE	0x1	PLLP_OUT5_OVRRIDE: 0 = disallow PLLP_OUT5 ratio override, 1 = enable override. 0 = DISABLE 1 = ENABLE
17	ENABLE	NONE	PLLP_OUT5_CLKEN: PLLP_OUT5 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
16	RESET_DISABLE	NONE	PLLP_OUT5_RSTN: PLLP_OUT5 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.2.258 CLK_RST_CONTROLLER_PLLP_MISC1_0

Offset: 0x680 | Read/Write: R/W | Reset: 0x30000000 (0bx011xxxx00000000000000000000000)

Bit	Reset	Description
30	0x0	PLLP_OUT5_DIV_BYP: 1 = bypass PLLP_OUT5 divider. Not glitchless. Make CLKEN bit 0 before toggling DIV_BYP bit to avoid glitch.
29	ENABLE	PLLP_HSIO_CLK_EN: Clock gate control for PLLP clock branch to HSIO. 1 = enable, 0 = disable 0 = DISABLE 1 = ENABLE
28	ENABLE	PLLP_XUSB_CLK_EN: Clock gate control for PLLP clock branch to XUSB. 1 = enable, 0 = disable 0 = DISABLE 1 = ENABLE
23:0	0x0	PLLP_SETUP: PLLP base setup bits

5.2.259 CLK_RST_CONTROLLER_EMC_DIV_CLK_SHAPER_CTRL_0

Note: EMC_PLLC_SHAPER_CTRL is a misnomer. It affects PLLP low jitter output, not PLLC.

EMC_DIV_CLK_SHAPER_CTRL and EMC_PLLC_SHAPER_CTRL shaper control registers should only be changed when the shaper paths are not being selected

EMC_DIV_CLK_SHAPER_CTRL shaper path is selected when ENABLE_EMC_DIV_SHAPER is set and EMC_2X_CLK_SRC is not 111

EMC_PLLC_SHAPER_CTRL shaper is selected when ENABLE_EMC_PLLC_SHAPER is set and EMC_2X_CLK_SRC is 111.

Offset: 0x68c | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_EMC_DIV_SHAPER: DISABLE: Not using the shaped clock for int_div_clk in the EMC switch 0 = DISABLE 1 = ENABLE
11	0x1	EMC_DIV_CLK_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
10	0x1	EMC_DIV_CLK_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
9	0x0	EMC_DIV_CLK_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter

Bit	Reset	Description
8	0x0	EMC_DIV_CLK_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
7	DEFAULT_SETTING	EMC_DIV_CLK_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins (RTL): default_values[5:0]. SW_PROG: Controlled by Software programmable register EMC_DIV_CLK_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	EMC_DIV_CLK_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S -> Y path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	EMC_DIV_CLK_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: Delay added on CLKIN(fall) -> CLKOUT(fall) Based on the EMC_DIV_CLK_SHAPER_CTRL[4:3] register 0 = DISABLE 1 = ENABLE
4:3	0x0	EMC_DIV_CLK_SHAPER_CTRL_FALL_DELAY
2	DISABLE	EMC_DIV_CLK_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: Delay added on CLKIN(rise) -> CLKOUT(rise) Controlled by the EMC_DIV_CLK_SHAPER_CTRL[1:0] register. 0 = DISABLE 1 = ENABLE
1:0	0x0	EMC_DIV_CLK_SHAPER_CTRL_RISE_DELAY

5.2.260 CLK_RST_CONTROLLER_EMC_PLLC_SHAPER_CTRL_0

Note: EMC_PLLC_SHAPER_CTRL is a misnomer. It affects PLLP low jitter output, not PLLC.

EMC_DIV_CLK_SHAPER_CTRL and EMC_PLLC_SHAPER_CTRL shaper control registers should only be changed when the shaper paths are not being selected

EMC_DIV_CLK_SHAPER_CTRL shaper path is selected when ENABLE_EMC_DIV_SHAPER is set and EMC_2X_CLK_SRC is not 111

EMC_PLLC_SHAPER_CTRL shaper is selected when ENABLE_EMC_PLLC_SHAPER is set and EMC_2X_CLK_SRC is 111

Offset: 0x690 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_EMC_PLLC_SHAPER: DISABLE: Not using the shaped clock for PLLP in the EMC switch 0 = DISABLE 1 = ENABLE
11	0x1	EMC_PLLC_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
10	0x1	EMC_PLLC_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
9	0x0	EMC_PLLC_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
8	0x0	EMC_PLLC_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than the delay of the inverter
7	DEFAULT_SETTING	EMC_PLLC_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins (RTL): default_values[5:0]. SW_PROG: Controlled by Software programmable register EMC_PLLC_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG

Bit	Reset	Description
6	ACTIVE	EMC_PLLC_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	EMC_PLLC_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: Delay added on CLKIN(fall) -> CLKOUT(fall) based on the EMC_PLLC_SHAPER_CTRL[4:3] register 0 = DISABLE 1 = ENABLE
4:3	0x0	EMC_PLLC_SHAPER_CTRL_FALL_DELAY
2	DISABLE	EMC_PLLC_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: Delay added on CLKIN(rise) -> CLKOUT(rise) Controlled by the EMC_PLLC_SHAPER_CTRL[1:0] register 0 = DISABLE 1 = ENABLE
1:0	0x0	EMC_PLLC_SHAPER_CTRL_RISE_DELAY

5.2.261 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0

Offset: 0x694 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Software Default	Description
31:29	PLLP_OUT3	PLLP_OUT0	SDMMC_LEGACY_TM_CLK_SRC: 000 = pllP_out3, 001 = pllC_out0, 010 = pllC2_out, 011 = CLK_M, 100 = pllP_out0, 101 = pllC4_out0, 110 = pllC4_out1, 111 = pllC4_out2 0 = PLLP_OUT3 1 = PLLC_OUT0 2 = PLLC2_OUT0 3 = CLK_M 4 = PLLP_OUT0 5 = PLLC4_OUT0 6 = PLLC4_OUT1 7 = PLLC4_OUT2
7:0	0x0	0x42	SDMMC_LEGACY_TM_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.262 CLK_RST_CONTROLLER_CLK_SOURCE_NVDEC_0

Offset: 0x698 | Read/Write: R/W | Reset: 0xe0000000 (0b111xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	NVDEC_CLK_SRC: 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA1_out0, 111 = CLK_M 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA1_OUT0 7 = CLK_M
7:0	0x0	NVDEC_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.263 CLK_RST_CONTROLLER_CLK_SOURCE_NVJPG_0

Offset: 0x69c | Read/Write: R/W | Reset: 0xe0000000 (0b111xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	NVJPG_CLK_SRC: 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA1_out0, 111 = CLK_M 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA1_OUT0 7 = CLK_M
7:0	0x0	NVJPG_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.264 CLK_RST_CONTROLLER_CLK_SOURCE_NVENC_0

Offset: 0x6a0 | Read/Write: R/W | Reset: 0x80000000 (0b100xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLP_OUT0	NVENC_CLK_SRC: 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA1_out0, 111 = CLK_M 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA1_OUT0 7 = CLK_M
7:0	0x0	NVENC_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.265 CLK_RST_CONTROLLER_PLLA1_BASE_0

Offset: 0x6a4 | Read/Write: R/W | Reset: 0x0X108002 (0b000xxxx00001xx00100000xx00000010)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLA1_BYPASS: 0 = no bypass, 1 = bypass. "DIRECT" 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLA1_ENABLE: 0 = disable, 1 = enable. will invert and connect to IDDQ. "DIRECT" 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLA1_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. To go into the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLA1_FREQ_LOCK: 0 = not lock, 1 = frequency is locked.
26	RO	X	PLLA1_LOCK: 0 = not locked, 1 = is locked.
24:20	RW	0x1	PLLA1_DIVP: 0 = post divider (divide By N+1). “DIRECT”
17:10	RW	0x20	PLLA1_DIVN: PLL feedback divider. “LATCHED”
7:0	RW	0x2	PLLA1_DIVM: PLL input divider. “DIRECT”

5.2.266 CLK_RST_CONTROLLER_PLLA1_MISC_0_0

Offset: 0x6a8 | Read/Write: R/W | Reset: 0x40000000 (0bx1xxxxxxxxxx00000000000000000x00)

Bit	Reset	Software Default	Description
30	0x1	NONE	PLLA1_RESET: Reset for digital logic of the PLL - 0 = Out of reset, 1 = Reset asserted. “DIRECT” 0 = DISABLE 1 = ENABLE
19:4	0x0	0x8000	PLLA1_EXT_FRU

Bit	Reset	Software Default	Description
3	0x0	NONE	PLLA1_PTS: Base PLLA1 test output select. 0 = PTO is 0 1 = PTO is FO 1 = FO
1:0	0x0	NONE	PLLA1_LOOP_CTRL:

5.2.267 CLK_RST_CONTROLLER_PLLA1_MISC_1_0

Offset: 0x6ac | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1xxxxxxxxxxxxx0000xx00000000)

Bit	Reset	Description
27	0x1	PLLA1_IDDQ: 0 = OFF 1 = ON
13:10	0x0	PLLA1_EXT_SUBINT
7:0	0x0	PLLA1_DIVN_FRAC

5.2.268 CLK_RST_CONTROLLER_PLLA1_MISC_2_0

Offset: 0x6b0 | Read/Write: R/W | Reset: 0x1f720005 (0b00011111011100100000000xxx0x101)

Bit	Reset	Description
31:24	0x1f	PLLA1_PLL_LD_MEM
23:20	0x7	PLLA1_PLL_FRUG_HIGH
19:16	0x2	PLLA1_PLL_FRUG_LOW
15:8	0x0	PLLA1_FLL_LD_MEM
4	0x0	PLLA1_FLL_DIV
2:0	0x5	PLLA1_FLL_FRUG

5.2.269 CLK_RST_CONTROLLER_PLLA1_MISC_3_0

Offset: 0x6b4 | Read/Write: R/W | Reset: 0x000000c4 (0bxxxxxxxx00000000000000011000100)

Bit	Reset	Description
23:8	0x0	PLLA1_SETUP
7:6	0x3	PLLA1_KP
5:4	0x0	PLLA1_LDIV
3:0	0x4	PLLA1_PLL_LD_TOL

5.2.270 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_DMIC3_0

Audio sync clock source select

Offset: 0x6b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK. Also
3:0	0x0	SYNC_CLK_RATE: 0000 = reserved 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = I2S3 bit clock. 0100 = I2S4 bit clock. 0101 = I2S5 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = reserved 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5 6 = PLLA_OUT0 7 = EXT_VIMCLK

5.2.271 CLK_RST_CONTROLLER_CLK_SOURCE_DMIC3_0

Offset: 0x6bc | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLA_OUT0	DMIC3_CLK_SRC: 000 = pllA_out0, 001 = audio SYNC_CLK 1x or 2x, 010 = pllP_out0, 011 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK 2 = PLLP_OUT0 3 = CLK_M
7:0	0x0	DMIC3_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.272 CLK_RST_CONTROLLER_CLK_SOURCE_APE_0

Offset: 0x6c0 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	APE_CLK_SRC: 000 = pllA_out0, 001 = pllC4_out0, 010 = pllC_out0, 011 = pllC4_out1, 100 = pllP_out0, 101 = pllC4_out2, 110 = clk_m 0 = PLLA_OUT0 1 = PLLC4_OUT0 2 = PLLC_OUT0 3 = PLLC4_OUT1 4 = PLLP_OUT0 6 = CLK_M 5 = PLLC4_OUT2
7:0	0x0	APE_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.273 CLK_RST_CONTROLLER_CLK_SOURCE_QSPI_0

Offset: 0x6c4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31:29	CLK_M	QSPI_CLK_SRC: 000 = pllP_out0, 001 = pllC_out1, 010 = pllC_out0, 100 = pllC4_out2, 101 = pllC4_out1, 110 = CLK_M, 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLC_OUT1 2 = PLLC_OUT0 4 = PLLC4_OUT2 5 = PLLC4_OUT1 6 = CLK_M 7 = PLLC4_OUT0
8	0x0	QSPI_CLK_DIV2_SEL: 0 = DIV1 SDR mode, 1 = DIV2 DDR interface mode.
7:0	0x0	QSPI_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.274 CLK_RST_CONTROLLER_CLK_SOURCE_VI_I2C_0

Offset: 0x6c8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	VI_I2C_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 6 = CLK_M
15:0	0x0	VI_I2C_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.2.275 CLK_RST_CONTROLLER_CLK_SOURCE_USB2_HSIC_TRK_0

Offset: 0x6cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Software Default	Description
7:0	0x0	0x6	USB2_HSIC_TRK_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.276 CLK_RST_CONTROLLER_CLK_SOURCE_PEX_SATA_USB_RX_BYP_0

Offset: 0x6d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000xxxxxxx000000000)

Bit	Reset	Description
24	0x0	SATA_USB_PAD_RX_BYP_REFCLK_CE: CLOCK Enable
23:16	0x0	SATA_USB_PAD_RX_BYP_REFCLK_DIVISOR: Divide by [(N/2)+1]
8	0x0	PEX_USB_PAD_RX_BYP_REFCLK_CE: CLOCK Enable
7:0	0x0	PEX_USB_PAD_RX_BYP_REFCLK_DIVISOR: Divide by [(N/2)+1]

5.2.277 CLK_RST_CONTROLLER_CLK_SOURCE_MAUD_0

Offset: 0x6d4 | Read/Write: R/W | Reset: 0x40000000 (0b010xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	MAUD_CLK_SRC: 000 = pllP_out0, 001 = pllP_out3, 010 = clk_m, 011 = clk_s, 100 = pllA_out0 0 = PLLP_OUT0 1 = PLLP_OUT3 2 = CLK_M 3 = CLK_S 4 = PLLA_OUT0
7:0	0x0	MAUD_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.2.278 CLK_RST_CONTROLLER_CLK_SOURCE_TSECB_0

Offset: 0x6d8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	TSECB_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 101 = pllA1_out0, 110 = CLK_M (osc), 111 = pllC4_out0 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 5 = PLLA1_OUT0 6 = CLK_M 7 = PLLC4_OUT0
7:0	0x0	TSECB_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.279 CLK_RST_CONTROLLER_CLK_CPUG_MISC1_0

Offset: 0x6dc | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CPUG_CNTCLK_RATIO: Clock divider ratio for the CNTCLK clocks in CCPLEX 000 = div-by-1. 001 = div-by-2. 010 = div-by-3. 011 = div-by-4. 100 = div-by-5. 101 = div-by-6. Other values - Reserved

5.2.280 CLK_RST_CONTROLLER_ACLK_BURST_POLICY_0

Offset: 0x6e0 | Read/Write: R/W | Reset: 0x10006666 (0b00010000xxxxxxxx0110011001100110)

Bit	Reset	Description
31:28	0x1	ADSP_STATE: 0000=clk_m Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	0x0	COP_AUTO_AWAKEUP_FROM_FIQ: 0 = NOP: 1=Burst on COP FIQ
26	0x0	CPU_AUTO_AWAKEUP_FROM_FIQ: 0 = NOP; 1=Burst on CPU FIQ
25	0x0	COP_AUTO_AWAKEUP_FROM_IRQ: 0 = NOP; 1=Burst on COP IRQ
24	0x0	CPU_AUTO_AWAKEUP_FROM_IRQ: 0 = NOP; 1=Burst on CPU IRQ
15:12	0x6	AWAKEUP_FIQ_SOURCE: 0000 = pllA1_out0, 0001 = pllC_out0, 0010 = pllP_out0, 0011 = pllA_out0, 0100 = pllC2_out0, 0101 = pllC3_out0, 0110 = clk_m, 0111 = pllA_out, 1000 = pllA1_out0, (Bypass divider - low jitter), 1111 = pllA_out, (Bypass divider - low jitter) 0 = PLLA1_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M 7 = PLLA_OUT 8 = PLLA1_OUT_LJ 15 = PLLA_OUT_LJ
11:8	0x6	AWAKEUP_IRQ_SOURCE: Same definitions as AWAKEUP_FIQ_SOURCE 0 = PLLA1_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M 7 = PLLA_OUT 8 = PLLA1_OUT_LJ 15 = PLLA_OUT_LJ
7:4	0x6	AWAKEUP_RUN_SOURCE: Same definitions as AWAKEUP_FIQ_SOURCE 0 = PLLA1_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M 7 = PLLA_OUT 8 = PLLA1_OUT_LJ 15 = PLLA_OUT_LJ
3:0	0x6	AWAKEUP_IDLE_SOURCE: Same definitions as AWAKEUP_FIQ_SOURCE 0 = PLLA1_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0 4 = PLLC2_OUT0 5 = PLLC3_OUT0 6 = CLK_M 7 = PLLA_OUT 8 = PLLA1_OUT_LJ 15 = PLLA_OUT_LJ

5.2.281 CLK_RST_CONTROLLER_SUPER_ACLK_DIVIDER_0

Offset: 0x6e4 | Read/Write: R/W | Reset: 0x00000000 (0b0xx00000000000000000000000000000)

Bit	Reset	Description
31	DISABLE	SUPER_ADIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
28	DISABLE	ACLK_INVERT_DCD: See 'Invert Duty-Cycle Distortion control'. 1 = Enable Inversion of DCD (duty-cycle distortion) 0 = DISABLE 1 = ENABLE
27	0x0	SUPER_ADIV_DIS_FROM_COP_FIQ: 0 = COP FIQ doesn’t impact super clock divider enable. 1 = Disable super clock divider on COP FIQ.
26	0x0	SUPER_ADIV_DIS_FROM_CPU_FIQ: 0 = CPU FIQ doesn’t impact super clock divider enable. 1 = Disable super clock divider on CPU FIQ.
25	0x0	SUPER_ADIV_DIS_FROM_COP_IRQ: 0 = COP IRQ doesn’t impact super clock divider enable. 1 = Disable super clock divider on COP IRQ.
24	0x0	SUPER_ADIV_DIS_FROM_CPU_IRQ: 0 = CPU IRQ doesn’t impact super clock divider enable. 1 = Disable super clock divider on CPU IRQ.
23:16	0x0	ACLK_CLK_DIVISOR: Divide by [(N/2)+1]
15:8	0x0	SUPER_ADIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_ADIV_DIVISOR: Actual value = n + 1.

5.2.282 CLK_RST_CONTROLLER_NVENC_SUPER_CLK_DIVIDER_0

Offset: 0x6e8 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_NVENC_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_NVENC_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_NVENC_DIV_DIVISOR: Actual value = n + 1.

5.2.283 CLK_RST_CONTROLLER_VI_SUPER_CLK_DIVIDER_0

Offset: 0x6ec | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_VI_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_VI_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_VI_DIV_DIVISOR: Actual value = n + 1.

5.2.284 CLK_RST_CONTROLLER_VIC_SUPER_CLK_DIVIDER_0

Offset: 0x6f0 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_VIC_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_VIC_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_VIC_DIV_DIVISOR: Actual value = n + 1.

5.2.285 CLK_RST_CONTROLLER_NVDEC_SUPER_CLK_DIVIDER_0

Offset: 0x6f4 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_NVDEC_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_NVDEC_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_NVDEC_DIV_DIVISOR: Actual value = n + 1.

5.2.286 CLK_RST_CONTROLLER_ISP_SUPER_CLK_DIVIDER_0

Offset: 0x6f8 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_ISP_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_ISP_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_ISP_DIV_DIVISOR: Actual value = n + 1.

5.2.287 CLK_RST_CONTROLLER_ISPB_SUPER_CLK_DIVIDER_0

Offset: 0x6fc | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_ISPB_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_ISPB_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_ISPB_DIV_DIVISOR: Actual value = n + 1.

5.2.288 xCLK_RST_CONTROLLER_NVJPG_SUPER_CLK_DIVIDER_0

Offset: 0x700 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_NVJPG_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_NVJPG_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_NVJPG_DIV_DIVISOR: Actual value = n + 1.

5.2.289 CLK_RST_CONTROLLER_SE_SUPER_CLK_DIVIDER_0

Offset: 0x704 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_SE_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_SE_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_SE_DIV_DIVISOR: Actual value = n + 1.

5.2.290 CLK_RST_CONTROLLER_TSEC_SUPER_CLK_DIVIDER_0

Offset: 0x708 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_TSEC_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_TSEC_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_TSEC_DIV_DIVISOR: Actual value = n + 1.

5.2.291 CLK_RST_CONTROLLER_TSECB_SUPER_CLK_DIVIDER_0

Offset: 0x70c | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	ENABLE	SUPER_TSECB_DIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_TSECB_DIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_TSECB_DIV_DIVISOR: Actual value = n + 1.

5.2.292 CLK_RST_CONTROLLER_CLK_SOURCE_UARTAPE_0

Offset: 0x710 | Read/Write: R/W | Reset: 0xc0000002 (0b110xxxx0xxxxxxxx0000000000000010)

Bit	Reset	Description
31:29	CLK_M	UARTAPE_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 110 = clk_m UART baud clock divisors by default come directly from UART DLM/DLL registers in 16.0 format. enable the UART_DIV_ENB to use the 15.1 divisor below 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 6 = CLK_M
24	DISABLE	UARTAPE_DIV_ENB: 0=use UART DLM/DLL, 1=use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTAPE_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.293 CLK_RST_CONTROLLER_CLK_CPUG_MISC2_0

Offset: 0x714 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	0x0	CPUG_ATCLKPTM_RATIO: Clock divider ratio for the ATCLK_UPSIZER clocks in CCPLEX 000 = div-by-2. 001 = div-by-3. 010 = div-by-4. 011 = div-by-5. 100 = div-by-6. Other values - Reserved

5.2.294 CLK_RST_CONTROLLER_CLK_SOURCE_DBGAPB_0

Offset: 0x718 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DBGAPB_CLK_SRC: 010 = pllP_out0 110 = clk_m 2 = PLLP_OUT0 6 = CLK_M
7:0	0x0	DBGAPB_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.295 CLK_RST_CONTROLLER_CLK_CCPLEX_CC4_RET_CLK_ENB_0

Offset: 0x71c | Read/Write: R/W | Reset: 0x001fffff (0bxxxxxxxxxxxx11111111111111111111)

Bit	Reset	Description
20	ENABLE	JTAG_REG_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
19	ENABLE	SERDES4F_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
18	ENABLE	SOC_THERM_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
17	ENABLE	CLK_M_SYS_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
16	ENABLE	DBGAPB_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
15	ENABLE	CSITE_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
14	ENABLE	MC_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
13	ENABLE	MSELECT_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
12	ENABLE	LA_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
11	ENABLE	SCLK_SYS_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
10	ENABLE	DVFS_REF_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
9	ENABLE	DVFS_SOC_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
8	ENABLE	DBG_OSCOUT_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
7	ENABLE	CK32KHZ_IB_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
6	ENABLE	PLLP_OUT_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
5	ENABLE	PLLP_OUT4_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
4	ENABLE	CAR_T1CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
3	ENABLE	CAR_T4CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
2	ENABLE	CAR_T18CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	ENABLE	SPARE0_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE
0	ENABLE	SPARE1_CLK_CPU_CC4_RET_CE: 0 = DISABLE 1 = ENABLE

5.2.296 CLK_RST_CONTROLLER_ACTMON_CPU_CLK_0

Create a low frequency signal for the CPU Clock Activity Monitor.

Offset: 0x720 | Read/Write: R/W | Reset: 0x00000400 (0b0xxxxxxxxxxxxxxxxxxxx10000000000)

Bit	Reset	Description
31	DISABLE	ACTMON_CPU_CLK_ENB: 0 = DISABLE 1 = ENABLE
10:0	0x400	ACTMON_CPU_CLK_RATIO: Divide by N where N must be even (LSB bit 0 is ignored). As an exception, the N value of 0x000 will divide by 2048.

5.2.297 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_SAFE_0

Offset: 0x724 | Read/Write: R/W | Reset: 0x60000000 (0b011xx0xxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_2X_SAFE_CLK_SRC: 000 = pllM_out0: Clock path will be: PLLMA -> Switch -> Divider -> PLLM Bypass -> PLLM SYNC_CLKOUTP/N 001 = pllC_out0, 010 = pllP_out0, 011 = clk_m, 100 = pllM_out_for_emc: Clock path will be: PLLMA -> PLLM SYNC_CLKOUTP/N, 101 = pllMb_out_for_emc: Clock path will be: PLLMB -> PLLM SYNC_CLKOUTP/N, 110 = pllMb_out0: Clock path will be: PLLMB -> Switch -> Divider -> PLLM Bypass -> PLLM SYNC_CLKOUTP/N, 111 = PLLP_UD 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLMB_UD 6 = PLLMB_OUT0 7 = PLLP_UD
26	DISABLE	EMC_SAFE_INVERT_DCD:RESERVED' 0 = DISABLE 1 = ENABLE
7:0	0x0	EMC_2X_SAFE_CLK_DIVISOR: Divide by [(N/2)+1]

5.2.298 CLK_RST_CONTROLLER_SDMMC2_PLLC4_OUT0_SHAPER_CTRL_0

Offset: 0x728 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC2_PLLC4_OUT0_SHAPER: DISABLE: Not using the shaped clock for pllc4_out0 in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC2_PLLC4_OUT0_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC2_PLLC4_OUT0_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC2_PLLC4_OUT0_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC2_PLLC4_OUT0_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than delay of inverter

Bit	Reset	Description
7	DEFAULT_SETTING	SDMMC2_PLLC4_OUT0_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC2_PLLC4_OUT0_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	SDMMC2_PLLC4_OUT0_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC2_PLLC4_OUT0_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC2_PLLC4_OUT0_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC2_PLLC4_OUT0_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC2_PLLC4_OUT0_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC2_PLLC4_OUT0_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC2_PLLC4_OUT0_SHAPER_CTRL_RISE_DELAY

5.2.299 CLK_RST_CONTROLLER_SDMMC2_PLLC4_OUT1_SHAPER_CTRL_0

Offset: 0x72c | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC2_PLLC4_OUT1_SHAPER: DISABLE: Not using the shaped clock for pllc4_out1 in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC2_PLLC4_OUT1_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC2_PLLC4_OUT1_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC2_PLLC4_OUT1_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC2_PLLC4_OUT1_SHAPER_P1: Adjust Duty Cycle in step sizes smaller that delay of inverter
7	DEFAULT_SETTING	SDMMC2_PLLC4_OUT1_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC2_PLLC4_OUT1_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	SDMMC2_PLLC4_OUT1_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC2_PLLC4_OUT1_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC2_PLLC4_OUT1_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC2_PLLC4_OUT1_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC2_PLLC4_OUT1_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC2_PLLC4_OUT1_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC2_PLLC4_OUT1_SHAPER_CTRL_RISE_DELAY

5.2.300 CLK_RST_CONTROLLER_SDMMC2_PLLC4_OUT2_SHAPER_CTRL_0

Offset: 0x730 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC2_PLLC4_OUT2_SHAPER: DISABLE: Not using the shaped clock for pllc4_out2 in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC2_PLLC4_OUT2_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC2_PLLC4_OUT2_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC2_PLLC4_OUT2_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC2_PLLC4_OUT2_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than delay of inverter
7	DEFAULT_SETTING	SDMMC2_PLLC4_OUT2_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC2_PLLC4_OUT2_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	SDMMC2_PLLC4_OUT2_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC2_PLLC4_OUT2_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC2_PLLC4_OUT2_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC2_PLLC4_OUT2_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC2_PLLC4_OUT2_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC2_PLLC4_OUT2_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC2_PLLC4_OUT2_SHAPER_CTRL_RISE_DELAY

5.2.301 CLK_RST_CONTROLLER_SDMMC2_DIV_CLK_SHAPER_CTRL_0

Offset: 0x734 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC2_DIV_CLK_SHAPER: DISABLE: Not using the shaped clock for div_clk in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC2_DIV_CLK_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC2_DIV_CLK_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC2_DIV_CLK_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC2_DIV_CLK_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than delay of inverter
7	DEFAULT_SETTING	SDMMC2_DIV_CLK_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC2_DIV_CLK_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG

Bit	Reset	Description
6	ACTIVE	SDMMC2_DIV_CLK_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC2_DIV_CLK_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC2_DIV_CLK_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC2_DIV_CLK_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC2_DIV_CLK_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC2_DIV_CLK_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC2_DIV_CLK_SHAPER_CTRL_RISE_DELAY

5.2.302 CLK_RST_CONTROLLER_SDMMC4_PLLC4_OUT0_SHAPER_CTRL_0

Offset: 0x738 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC4_PLLC4_OUT0_SHAPER: DISABLE: Not using the shaped clock for pllc4_out0 in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC4_PLLC4_OUT0_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC4_PLLC4_OUT0_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC4_PLLC4_OUT0_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC4_PLLC4_OUT0_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than delay of inverter
7	DEFAULT_SETTING	SDMMC4_PLLC4_OUT0_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC4_PLLC4_OUT0_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	SDMMC4_PLLC4_OUT0_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC4_PLLC4_OUT0_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC4_PLLC4_OUT0_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC4_PLLC4_OUT0_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC4_PLLC4_OUT0_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC4_PLLC4_OUT0_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC4_PLLC4_OUT0_SHAPER_CTRL_RISE_DELAY

5.2.303 CLK_RST_CONTROLLER_SDMMC4_PLLC4_OUT1_SHAPER_CTRL_0

Offset: 0x73c | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC4_PLLC4_OUT1_SHAPER: DISABLE: Not using the shaped clock for pllc4_out1 in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC4_PLLC4_OUT1_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC4_PLLC4_OUT1_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC4_PLLC4_OUT1_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC4_PLLC4_OUT1_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than delay of inverter
7	DEFAULT_SETTING	SDMMC4_PLLC4_OUT1_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC4_PLLC4_OUT1_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	SDMMC4_PLLC4_OUT1_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC4_PLLC4_OUT1_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC4_PLLC4_OUT1_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC4_PLLC4_OUT1_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC4_PLLC4_OUT1_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC4_PLLC4_OUT1_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC4_PLLC4_OUT1_SHAPER_CTRL_RISE_DELAY

5.2.304 CLK_RST_CONTROLLER_SDMMC4_PLLC4_OUT2_SHAPER_CTRL_0

Offset: 0x740 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC4_PLLC4_OUT2_SHAPER: DISABLE: Not using the shaped clock for pllc4_out2 in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC4_PLLC4_OUT2_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC4_PLLC4_OUT2_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC4_PLLC4_OUT2_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC4_PLLC4_OUT2_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than delay of inverter
7	DEFAULT_SETTING	SDMMC4_PLLC4_OUT2_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC4_PLLC4_OUT2_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG

Bit	Reset	Description
6	ACTIVE	SDMMC4_PLLC4_OUT2_SHAPER_CTRL_SW_BYPASS: ACTIVE : Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC4_PLLC4_OUT2_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC4_PLLC4_OUT2_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC4_PLLC4_OUT2_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC4_PLLC4_OUT2_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC4_PLLC4_OUT2_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC4_PLLC4_OUT2_SHAPER_CTRL_RISE_DELAY

5.2.305 CLK_RST_CONTROLLER_SDMMC4_DIV_CLK_SHAPER_CTRL_0

Offset: 0x744 | Read/Write: R/W | Reset: 0x00000c00 (0b0xxxxxxxxxxxxxxxxxxx110000000000)

Bit	Reset	Description
31	DISABLE	ENABLE_SDMMC4_DIV_CLK_SHAPER: DISABLE: Not using the shaped clock for div_clk in SDMMC switch 0 = DISABLE 1 = ENABLE
11	0x1	SDMMC4_DIV_CLK_SHAPER_N0: Adjust Duty Cycle in step sizes smaller than delay of inverter
10	0x1	SDMMC4_DIV_CLK_SHAPER_N1: Adjust Duty Cycle in step sizes smaller than delay of inverter
9	0x0	SDMMC4_DIV_CLK_SHAPER_P0: Adjust Duty Cycle in step sizes smaller than delay of inverter
8	0x0	SDMMC4_DIV_CLK_SHAPER_P1: Adjust Duty Cycle in step sizes smaller than delay of inverter
7	DEFAULT_SETTING	SDMMC4_DIV_CLK_SHAPER_CTRL_TRIM_SELECT: DEFAULT_SETTING: Controlled by shaper pins(RTL): default_values[5:0]. SW_PROG: Controlled by S/W Programmable Register SDMMC4_DIV_CLK_SHAPER_CTRL[5:0] 0 = DEFAULT_SETTING 1 = SW_PROG
6	ACTIVE	SDMMC4_DIV_CLK_SHAPER_CTRL_SW_BYPASS: ACTIVE: Shaper circuit in clock path BYPASS: Shaper Bypassed. S->Y Path to reduce distortion 0 = ACTIVE 1 = BYPASS
5	DISABLE	SDMMC4_DIV_CLK_SHAPER_CTRL_DF: DISABLE: No delay on CLKIN(fall) -> CLKOUT(fall) ENABLE: delay Added on CLKIN(fall) -> CLKOUT(fall) based on Register SDMMC4_DIV_CLK_SHAPER_CTRL[4:3] 0 = DISABLE 1 = ENABLE
4:3	0x0	SDMMC4_DIV_CLK_SHAPER_CTRL_FALL_DELAY
2	DISABLE	SDMMC4_DIV_CLK_SHAPER_CTRL_DR: DISABLE: No delay on CLKIN(rise) -> CLKOUT(rise) ENABLE: delay Added on CLKIN(rise) -> CLKOUT(rise) Controlled by Register SDMMC4_DIV_CLK_SHAPER_CTRL[1:0] 0 = DISABLE 1 = ENABLE
1:0	0x0	SDMMC4_DIV_CLK_SHAPER_CTRL_RISE_DELAY

CHAPTER 6: CL-DVFS

The CL_DVFS module contains control logic that provides a hardware mechanism for controlling the clock rate and power supply voltage of the fast CPU complex.

This chapter describes the closed-loop dynamic voltage and frequency scaling (CL-DVFS) registers. It is not intended to be a programming guide to CL-DVFS, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

6.1 CL-DVFS Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

6.1.1 CL_DVFS_CTRL_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	DFLL_CTRL_MODE: DFLL Control Mode. 0 = DISABLE: Disabled. The ring oscillator does not run. 1 = ENABLE_OPEN_LOOP: In Enable Open Loop mode, the ring oscillator will run, but the control loop will remain inactive. 2 = ENABLE_CLOSED_LOOP: In Enable Closed Loop mode, the control loop is enabled and the I2C interface will continuously update the control output in order to maintain the desired frequency set in the DFLL_FREQ_REQ_xxxx fields.

6.1.2 CL_DVFS_CONFIG_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000110)

Bit	Reset	Description
7:0	0x6	DFLL_CONFIG_DIV_S: Refclk divider for setting DFLL control loop sample rate. Refclk is divided by 32x the value in this field

6.1.3 CL_DVFS_PARAMS_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x000000f0 (0bxxxxxxx000000000xxxxx00011110000)

Bit	Reset	Description
24	0x0	DFLL_PARAMS_CG_SCALE: If set to 1, reduces the overall loop gain by a factor of 8.
23:22	0x0	DFLL_PARAMS_FORCE_MODE: 0 = DISABLE: Disabled. The I2C control value is never forced during a frequency change. 1 = FIXED: In Fixed Delay mode, the I2C control value is forced for a fixed number of sample periods. 2 = AUTO: In Auto mode, the I2C control value is forced for a calculated number of sample periods.
21:16	0x0	DFLL_PARAMS_CF_PARAM: Length of time that a forced I2C control value will be applied after a frequency change if forcing was requested for that change. In Fixed Delay mode, it provides the number of sample periods to force the I2C control value. In Auto mode, the force time is equal to I2C control output delta * cf_param / 16

Bit	Reset	Description
10:8	0x0	DFLL_PARAMS_CI_PARAM: Integral term gain in the control loop controls how a cycle deficit/surfeit after a frequency change is cleared. It temporarily raises or lowers the core voltage to allow the total clock cycle count to converge with the ideal number of cycles that should have been produced since the last frequency change request. 0 = DISABLE 1 = DIV2 2 = DIV4 3 = DIV8 4 = DIV16 5 = DIV32 6 = DIV64 7 = DIV128
7:0	0xf0	DFLL_PARAMS_CG_PARAM: Overall loop gain control (SIGNED value), controls the overall response time of the control loop

6.1.4 CL_DVFS_TUNE0_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DFLL_TUNE0_DLY_STK: Drives I_DLY_SPARE<15:0>
15:8	0x0	DFLL_TUNE0_DLY_STK: input bits to both coarse (4) and fine (4) tune the delay of stacked gates like chain. Drives I_DLY_STK<7:0>.
7:0	0x0	DFLL_TUNE0_DLY_INV: Input bits to both coarse (3) and fine (5) tune the delay of the inverter path. Drives I_DLY_INV<7:0>.

6.1.5 CL_DVFS_TUNE1_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:23	0x0	DFLL_TUNE1_DLY_FINE: Input bits to tune the two phases of the clock. 8 bits to tune, and 1 bit to choose high vs. low. Drives I_DLY_FINE<8:0>.
22:12	0x0	DFLL_TUNE1_DLY_SRAM: input bits to both coarse (3) and fine (5) tune the delay of sram path like chain.Drives I_DLY_SRAM<10:0> of DVCO macro
11	0x0	DFLL_TUNE1_DLY_SPARE1: Drives I_DLY_SPARE<16> of DVCO macro
10:0	0x0	DFLL_TUNE1_DLY_WIRE: Input bits to both coarse (2) and fine (9) tune the delay of wire dominated path. Drives I_DLY_WIRE<10:0>.

6.1.6 CL_DVFS_FREQ_REQ_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000ff40 (0bxxx00000000000011111111101000000)

Bit	Reset	Description
28	0x0	DFLL_FREQ_REQ_FORCE_EN: Set to '1' to force I2C control output to initial value specified by the 'force_val' field
27:16	0x0	DFLL_FREQ_REQ_FORCE_VAL: Value forced onto the integrator during a frequency transition, only used if the 'force_en' field below is '1' AND the global 'force_mode' field of the dfll_params register is not set to 'disable'. The best value is (desired I2C control value - safe I2C control value) x 128) / Cg.
15:8	0xff	DFLL_FREQ_REQ_SCALE: Proportion of output clock cycles (+1) to *not* skip over a period of 256 cycles
7	0x0	DFLL_FREQ_REQ_VALID: Set to '1' to indicate that the frequency configuration is valid and should be used. If this bit is '0', the control loop will revert to 'open loop' mode, and the I2C control interface value will be determined by the 'safe' value.
6:0	0x40	DFLL_FREQ_REQ_MULT: Primary frequency multiplication factor 'F'. The ring oscillator output frequency will be (REF_CLK/2) x F.

6.1.7 CL_DVFS_SCALE_RAMP_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	DFLL_OUTPUT_RAMP_RATE: The ramp up/ramp down rate of the control signal to the output scaler. Determines the number of cycles (+1) to wait for each counter step.

6.1.8 CL_DVFS_DROOP_CTRL_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000810 (0bxxxxxxxx00000000xxxx100000010000)

Bit	Reset	Description
23:16	0x0	DFLL_DROOP_CTRL_MIN_FREQ: The minimum allowed ring oscillator frequency before the 'droop' clock skipper is enabled. The value written determines the minimum number of cycles that are allowed in 4 REF_CLK cycles' Fmin = droop_ctrl.min_freq * (REF_CLK / 4)
11:8	0x8	DFLL_DROOP_CTRL_CUT: CPU clock is scaled by (cut+1)/16 immediately after reaching the minimum ring oscillator frequency
7:0	0x10	DFLL_DROOP_CTRL_RATE: Controls the rate at which clock cycles are re-introduced to the droop skipper after it has been ramped down to compensate for a frequency droop. It is the number of cycles (+1) to wait for each counter step

6.1.9 CL_DVFS_OUTPUT_CFG_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x50200000 (0bx1010000xx100000xx00000000000000)

Bit	Reset	Description
30	0x1	DFLL_OUTPUT_CONFIG_I2C_ENABLE: Master enable control for I2C control value updates. If this field is '0', then I2C control messages are inhibited, regardless of the DFLL mode.
29:24	0x10	DFLL_OUTPUT_CONFIG_SAFE: 'Safe' value for the OUTPUT control interface. This value will be output whenever OUTPUT is enabled but the DFLL is disabled, or in open loop mode.
21:16	0x20	DFLL_OUTPUT_CONFIG_MAX: Maximum allowed value on the OUTPUT control interface.
13:8	0x0	DFLL_OUTPUT_CONFIG_MIN: Minimum allowed value on the OUTPUT control interface.
7	0x0	DFLL_OUTPUT_CONFIG_DELTA_EN: 1: In conjunction with 'clk_en'=1, causes the PWM clock/data output to only become active (for 32 cycles) whenever a change in the PWM value occurs. 0: The PWM data/clock outputs run continuously when enabled.
6	0x0	DFLL_OUTPUT_CONFIG_CLK_EN: Enables the PMIC control clock output for digitally controlled PMICs.
5:0	0x0	DFLL_OUTPUT_CONFIG_DIV_D: Divider setting for PWM PMIC control output (divides the SOC clock).

6.1.10 CL_DVFS_OUTPUT_FORCE_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6	0x0	DFLL_OUTPUT_FORCE_ENABLE: Enable the force value onto the OUTPUT control output
5:0	0x0	DFLL_OUTPUT_FORCE_VALUE: Value to force OUTPUT control output whenever the i2c_ctrl_force_enable field is set to 1.

6.1.11 CL_DVFS_MONITOR_CTRL_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	0x0	DFLL_MONITOR_CTRL_SELECT: Selects a control loop data source for monitoring 0 = DISABLE 1 = CYCLE_INT 2 = PRO_TERM 3 = INT_TERM 4 = OUTPUT_INT 5 = OUTPUT_VALUE 6 = FREQ

6.1.12 CL_DVFS_MONITOR_DATA_0

Offset: 0x2c | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	DFLL_MONITOR_DATA_NEW: Set to '1' whenever a new sample is generated. Cleared on reads.
15:0	X	DFLL_MONITOR_DATA_VAL: Read data monitor source selected by dfll_monitor_ctrl_select. If monitoring is enabled, this field is updated every sample period.

6.1.13 CL_DVFS_I2C_CFG_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x0010f000 (0bxxxxxxxxxxxx1x0001111x00000000000)

Bit	Reset	Description
20	0x1	I2C_BUS_ARB: Enables arbitration for bus
18:16	0x0	I2C_MASTER_CODE: Master code for high-speed transfers
15	0x1	I2C_PACKET_MODE: Enables Packet mode for high-speed transfers
14:12	0x7	I2C_SIZE: Size of voltage values from 1 to 7
10	0x0	I2C_ADDR_7BIT_10BIT: 0: Selects 7-bit addressing 1: Select 10-bit addressing
9:0	0x0	I2C_SLAVE_ID: External slave ID address

6.1.14 CL_DVFS_I2C_VDD_REG_ADDR_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	I2C_DEFAULT_DATA: Default data
7:0	0x0	I2C_ADDR_DATA: Address for voltage sel

6.1.15 CL_DVFS_I2C_STS_0

Offset: 0x48 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:1	X	I2C_LAST_VALUE: Output value from the DFLL of last I2C request that was completed successfully
0	X	I2C_REQ_PENDING: Indicates there is an outstanding I2C request

6.1.16 CL_DVFS_INTR_STS_0

Offset: 0x5c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MAX_VDD_LIMIT_INTR: Interrupt to indicate the DFLL has hit the VDD ceiling as programmed in DFLL_OUTPUT_CONFIG_MAX

Bit	Reset	Description
0	0x0	MIN_VDD_LIMIT_INTR: Interrupt to indicate the DFLL has hit the VDD floor as programmed in DFLL_OUTPUT_CONFIG_MIN

6.1.17 CL_DVFS_INTR_EN_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MAX_VDD_LIMIT_INTR_EN: Enable for interrupt to indicate the DFLL has hit the VDD ceiling as programmed in DFLL_OUTPUT_CONFIG_MAX
0	0x0	MIN_VDD_LIMIT_INTR_EN: Enable for interrupt to indicate the DFLL has hit the VDD floor as programmed in DFLL_OUTPUT_CONFIG_MIN

6.1.18 CL_DVFS_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- SCL frequency (Std/Fast/Fm+ modes) = ClkSourceFreq/((tlow+thigh+3)*(N+1)) for lower values of N, up to 3
- SCL frequency (Std/Fast/Fm+ modes) = ClkSourceFreq/((tlow+thigh+2)*(N+1)) for higher values of N, above 3
- SCL frequency (HS mode) = ClkSourceFreq/((ths_low+ths_high+4)*(N+1)) for lower values of N, up to 4
- SCL frequency (HS mode) = ClkSourceFreq/ ((ths_low+ths_high+2)*(N+1)) for higher values of N, above 4

Offset: 0x16c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000110010000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE:N = Divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE:N = Divide by n+1

6.1.19 CL_DVFS_OUTPUT_LUT_0

This lookup table (LUT) contains voltage LUT values stored in a 33-deep by 8 bit wide RAM. To program the LUT, simply sequentially address the RAM, using the base offset address of 0x20 (10’b10xxxxxxx). Addresses are DWORD aligned, not BYTE aligned. PMIC values must be programmed to span the entire desired operating range, and be monotonically increasing. The logic assumes that the middle entry (16th) corresponds to the ‘safe’ initial voltage, as specified in the DFLL_OUTPUT_CONFIG.SAFE field. DFLL_OUTPUT_CONFIG_SAFE is initialized by default to 0x10, which corresponds to the middle of the LUT. However, the safe value can point to any LUT entry.

6.1.20 DVFS_DFLL_THROTTLE_CTRL_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxx0000xxxx0110)

Bit	Reset	Description
11:9	0x0	DFLL_THROTTLE_OVERRIDE_INDEX: Index value for override
8	0x0	DFLL_THROTTLE_OVERRIDE_EN: Override soc2cldvfs throttling of CLDVFS. Aids debug by giving software direct control of the index
3	0x0	DFLL_THROTTLE_CONSTRAIN_MIN: If set to '1' the throttle constrains OUTPUT_CONFIG_MIN. This should be chosen if the PMIC has negative gain, because the maximum voltage is, in that case, represented by the minimum control value
2	0x1	DFLL_FAVOR_HW_ARB: Prioritize hardware I2C requests when throttling enabled
1	0x1	DFLL_THROTTLE_DISABLE_OUTPUT_FORCE: Set to '1' to disable software-controlled OUTPUT_FORCE during hardware-controlled voltage throttling
0	0x0	DFLL_THROTTLE_EN: Enable hardware-controlled voltage throttling

6.1.21 DVFS_DFLL_THROTTLE_LIGHT_0

Offset: 0x68 | Read/Write: R/W | Reset: 0xff402000 (0b11111111x1000000x0100000xxxxxxx0)

Bit	Reset	Description
31:24	0xff	LIGHT_SCALE: Skipper-2 control during light throttling
22:16	0x40	LIGHT_MULT: Frequency multiplier during light throttling
14:8	0x20	LIGHT_VSEL_MAX: Maximum voltage limit during light throttling
0	0x0	LIGHT_OVERRIDE: Set to '1' to override DFLL_THROTTLE_LIGHT frequency multiplier, frequency scale, and maximum voltage limit with software-requested values

6.1.22 DVFS_DFLL_THROTTLE_MEDIUM_0

Offset: 0x6c | Read/Write: R/W | Reset: 0xff402000 (0b11111111x1000000x0100000xxxxxxx0)

Bit	Reset	Description
31:24	0xff	MEDIUM_SCALE: Skipper-2 control during medium throttling
22:16	0x40	MEDIUM_MULT: Frequency multiplier during medium throttling
14:8	0x20	MEDIUM_VSEL_MAX: Maximum voltage limit during medium throttling
0	0x0	MEDIUM_OVERRIDE: Set to '1' to override DFLL_THROTTLE_MEDIUM frequency multiplier, frequency scale, frequency scale, and maximum voltage limit with software-requested values

6.1.23 DVFS_DFLL_THROTTLE_HEAVY_0

Offset: 0x70 | Read/Write: R/W | Reset: 0xff402000 (0b11111111x1000000x0100000xxxxxxx0)

Bit	Reset	Description
31:24	0xff	HEAVY_SCALE: Skipper-2 control during heavy throttling
22:16	0x40	HEAVY_MULT: Frequency multiplier during heavy throttling
14:8	0x20	HEAVY_VSEL_MAX: Maximum voltage limit during heavy throttling
0	0x0	HEAVY_OVERRIDE: Set to '1' to override DFLL_THROTTLE_HEAVY frequency multiplier, frequency scale, frequency scale, and maximum voltage limit with software-requested values

6.1.24 DVFS_CC4_HVC_0

Note that HVC is now referred to as CC3. See Chapter 16: CPU Complex for more information on CC3 and CC4.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000101 (0bxxxxxxxxxxxxxxxxxxxxxxxxx100000001)

Bit	Reset	Description
8	0x1	CC4_HVC_FORCE_ENABLE: Enable force value onto OUTPUT control output
7:2	0x0	CC4_HVC_FORCE_VALUE: Value to force OUTPUT control output whenever the i2c_ctrl_force_enable field is set to 1
1:0	0x1	CC4_HVC_DFLL_CTRL_MODE: 0 = DISABLE: 0 = DFLL Disabled. The ring oscillator does not run. 1 = ENABLE_OPEN_LOOP. The ring oscillator will run, but the control loop will remain inactive 2 = ENABLE_CLOSED_LOOP. The control loop is enabled, and the I2C interface will update the control output as needed in order to maintain the desired frequency set in the CLDVFS DFLL_FREQ_REQ_xxxx fields.

CHAPTER 7: ATOMICS

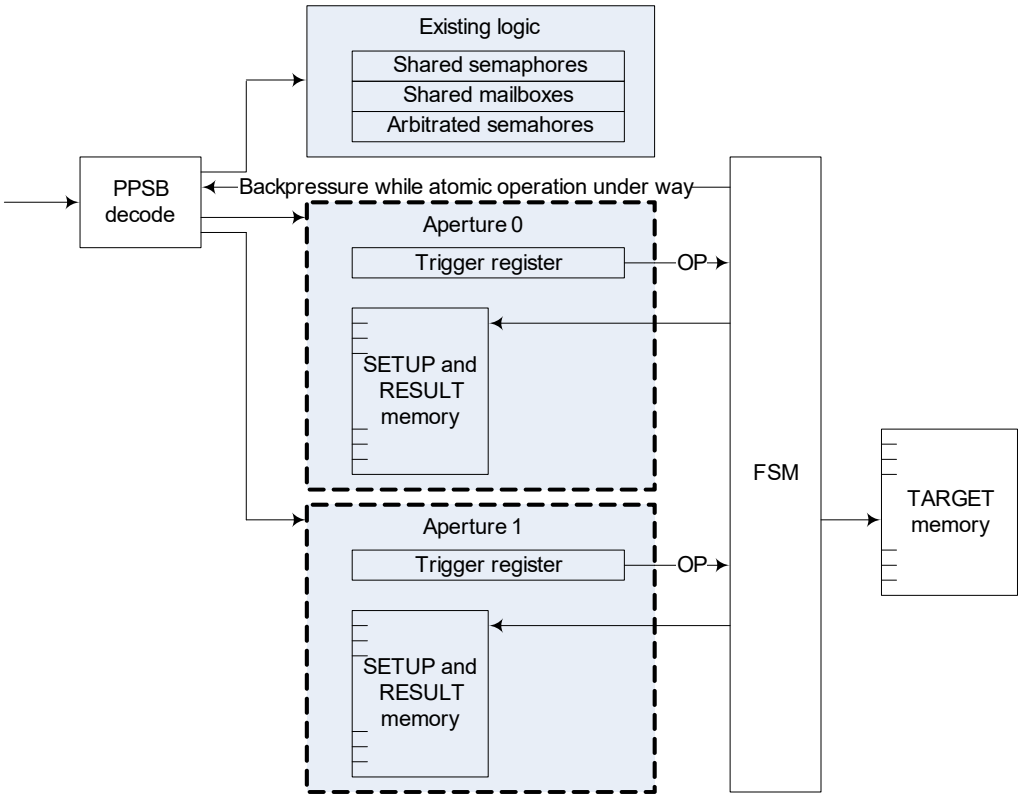
7.1 Introduction

The Atomics block assists software tasks with maintaining synchronization. It is parallel to the existing Semaphores block and intended to extend that block's functionality with new capabilities.

7.2 Functionality

The figure below shows a reference decomposition of the hardware synchronization elements in functional blocks.

Figure 8: Reference Block Diagram



7.3 Synchronization Elements

The synchronization elements are atomic primitives. Their main advantage is to allow some specific algorithms that access shared resources to be performed lock free, for better performance.

In general, the atomic primitives below have results equivalent to atomic RMW operations on a target register. The cycle is made atomic without using locked bus accesses as explained below.

7.3.1 General Principles

Atomic primitives maintain atomicity by splitting the whole operation in three phases:

- A setup phase where parameters are prepared in registers. There are as many sets of setup registers as there are apertures.

- A trigger register that starts a specific operation. The trigger register fits in one 32-bit word to ensure atomic issue of the operation. There is one trigger register per aperture. The trigger register specifies:
  - The primitive operation to be performed.
  - The index of the target register for the primitive operation.
- A result phase where the results of the operation are made available. There are as many sets of result registers as there are sets of setup registers. The operation, once triggered, is executed atomically.

The setup, result and trigger registers are duplicated in separate apertures, one aperture per client of the block. There are two such apertures in Tegra X1, normally one used by CPU, the other by BPMP-Lite, but this is not enforced by hardware as it is for the arbitrated semaphores.

Setup, target and result registers are 32-bits each with no further structure imposed by hardware. The trigger register is 32 bits, structured in command and target fields.

The description of the different operations is based on pseudo code that uses the following conventions:

- ATOMIC_OP identifies the exact operation and index ID identifies a given target register. Both ATOMIC_OP and ID are duplicated per aperture
- Index A identifies the aperture from which the operation is executed.
- The setup registers in a set are stored in arrays v and c. The result registers are stored in array old. All of these are duplicated per aperture.
- The target registers are not duplicated per aperture and are stored in an array T.
- The semantic of assignment is like C. All assignments are assumed to take place instantaneously. Intermediate values are used to get the equivalent of clocked operations of the hardware. Typically, the hardware implementation has no intermediate values if operations are performed in one clock cycle.
- Arithmetic operations are performed as signed 32 bits operations modulo 232 using 2s complement coding.

7.3.2 Atomic Exchange

The only purpose of an atomic exchange is to read a value and replace it with another value in an atomic fashion.[1]

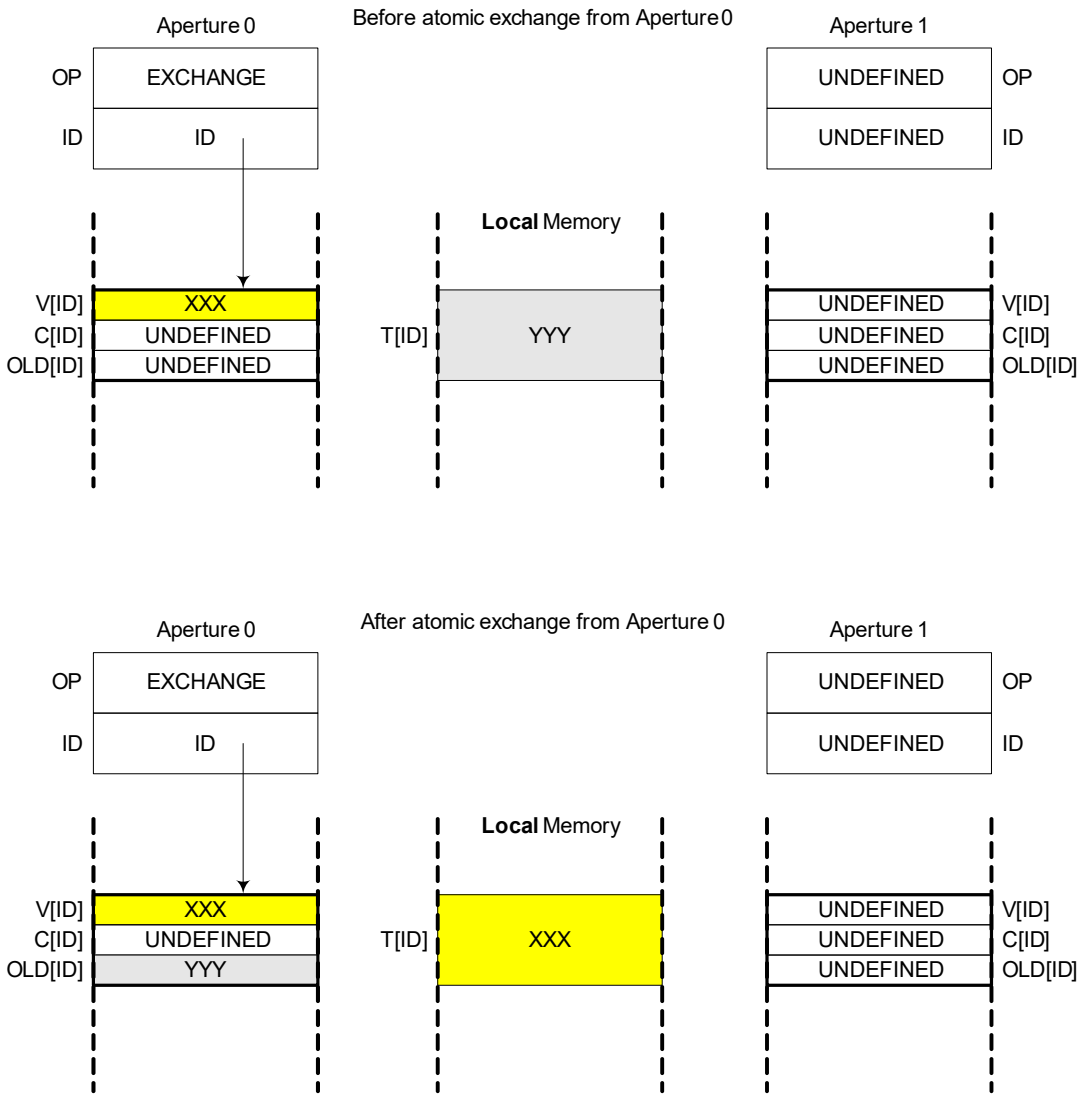
The corresponding pseudo code is

```
if (ATOMIC_OP[A] == EXCHANGE) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] = v[A][ID[A]]
}
```

The figure below illustrates an atomic exchange performed using aperture 0.

1. The result register always contains the old value of the target register, i.e. the value before applying the atomic operation. This is true even for PUT, so that PUT and EXCHANGE are semantically equivalent with the current definition.

Figure 9: Atomic Exchange Operation



7.3.3 Atomic Compare and Exchange

This is a variant of atomic exchange where the exchange only takes place if the old value matches the second setup register.

```
if (ATOMIC_OP[A] == COMPARE_AND_EXCHANGE) && (T[ID[A]] == c[A][ID[A}]) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] = v[A][ID[A]]
}
```

7.3.4 Atomic Increment

The atomic increment simply ensures that different processors can increment a shared value without losing any increment. The operation is labeled INCREMENT, but, for 32 bit operations, incrementing by 232 – k is equivalent to decrementing by k.

```
if (ATOMIC_OP[A] == INCREMENT) {
  old[A][ID[A]] = Target[ID[A]]
  T[ID[A]] += v[A][ID[A]]
}
```

The variant DECREMENT_WITH_ZERO_SATURATE is similar, adding a saturation to zero when going down. This may be used to manage shared resources that are present in finite numbers.

```
if (ATOMIC_OP[A] == DECREMENT_WITH_ZERO_SATURATE) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] -= v[A][ID[A]]
  if MSB(T[ID[A]]) == 1b && MSB(r[A][ID[A]]) == 0b { T[ID[A]] = 0 }
}
```

7.3.5 Atomic Get and Put

These are trivial for hardware if there is only one access port, in which case they act as normal register operations. They are nevertheless defined to allow more complex future implementations where the atomic operation blocks could present multiple hardware interfaces to the rest of the system.

```
if (ATOMIC_OP[A] == GET) {
  old[A][ID[A]] = T[ID[A]]
}

if (ATOMIC_OP[A] == PUT) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] = v[A][ID[A]]
}
```

7.3.6 Atomic Test and Set□Test and Clear□Test and Invert

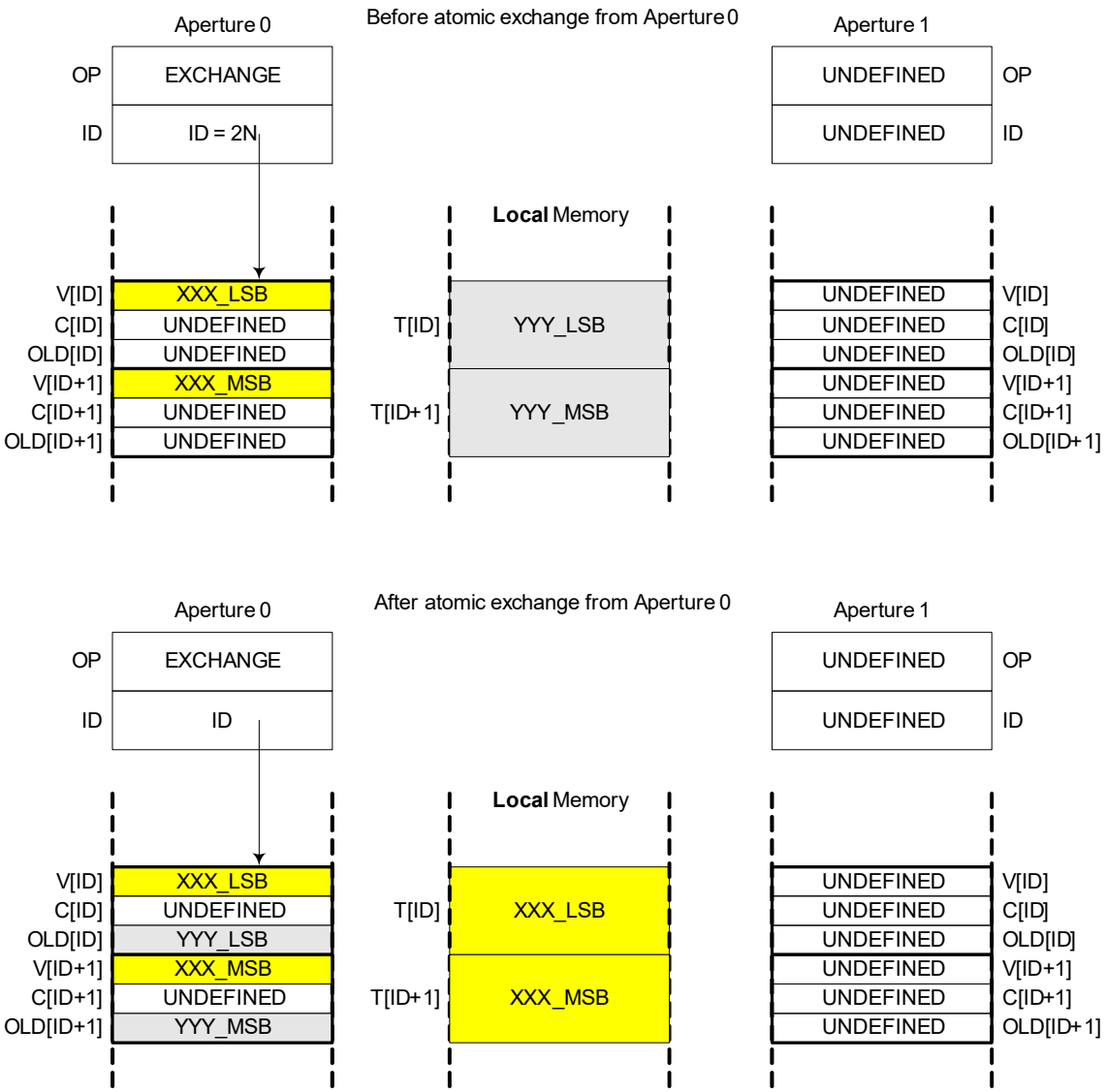
These are simple extensions of previous cases, where the operation is either a bit set, bit clear or bit invert.

```
if (ATOMIC_OP[A] == TEST_AND_SET) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] |= v[A][ID[A]]
}

if (ATOMIC_OP[A] == TEST_AND_CLEAR) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] &= ~v[A][ID[A]]
}

if (ATOMIC_OP[A] == TEST_AND_INVERT) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] ^= v[A][ID[A]]
}
```

7.3.7 Width of Operations

All operations can operate on 32 bits or 64 bits. 64 bits operations use an even/odd pair of address, with the MSB at the odd address. The next figure shows how a 64 bits exchange operates. Other operations behave similarly. In the case of atomic increment and decrement operations, the carry will correctly propagate from the LSB to the MSB.

Figure 10: 64-bit Atomic Exchange Operation



7.4 Atomics Registers

7.4.1 ATOMICS_AP0_TRIGGER_0

Trigger Registers

Offset: 000h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
22:16	X	ID
4	X	WIDTH64

Bit	Reset	Description
3:0	X	CMD: 0 = EXCHANGE 1 = COMPARE_AND_EXCHANGE 2 = INCREMENT 3 = DECREMENT_W_ZERO_SATURATE 4 = GET 5 = PUT 6 = TEST_AND_SET 7 = TEST_AND_CLEAR 8 = TEST_AND_INVERT

7.4.2 ATOMICS_AP1_TRIGGER_0

Offset: 004h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
22:16	X	ID
4	X	WIDTH64
3:0	X	CMD: 0 = EXCHANGE 1 = COMPARE_AND_EXCHANGE 2 = INCREMENT 3 = DECREMENT_W_ZERO_SATURATE 4 = GET 5 = PUT 6 = TEST_AND_SET 7 = TEST_AND_CLEAR 8 = TEST_AND_INVERT

7.4.3 ATOMICS_AP0_SETUP_V_0

Aperture 0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 400h..5ffh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	V

7.4.4 ATOMICS_AP0_SETUP_C_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 800h..9ffh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C

7.4.5 ATOMICS_AP0_RESULT_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: c00h..dffh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	R

7.4.6 ATOMICS_AP1_SETUP_V_0

Aperture 1

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 1000h..11ffh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	V

7.4.7 ATOMICS_AP1_SETUP_C_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 1400h..15ffh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	V

7.4.8 ATOMICS_AP1_RESULT_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 1800h..19ffh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	R

CHAPTER 8: TIMERS

This chapter documents the various timers available to software in a Tegra® X1 system. The following table summarizes these timers.

Table 19: List of Timers

Name	Primary Use	Related Interrupt	Secure	Freq.	Notes
RTC	Wall Clock Timer	RTC	Pseudo	32kHz	See Chapter 11: Real-Time Clock in this TRM for detailed information.
TMR	NVIDIA® Generic Timers	TMR9-0 TMR10-13	Cfg	1MHz OSC	Configurable to be secure
WDT	Per CPU/COP	WDT_<>	Cfg	1MHz	Configurable to be secure
TSC	Reference for GT	N/A	Yes	OSC	Counter value can only be updated in secure mode.
GT	ARM CPU Generic Timers	PPIs(1)	Yes	TSC	These timers use TSC as reference.

1. PPIs are per CPU Private Peripheral Interrupts

8.1 ARM CPU Generic Timers (GTs)

The ARM® Generic Timer architecture defines generic CPU timer requirements. The architecture requires that the SoC supply a timestamp counter reference that is independent of the CPU clock frequency. Cortex®-A57 generic timer events are not affected by CPU clock frequency change.

The CPU Generic Timer includes:

- A physical counter that contains the count value of the system counter.
- A virtual counter that indicates virtual time. The virtual counter contains the value of the physical counter minus a 64-bit virtual offset.
- A set of four timers per CPU.

The four timers are the following:

- Secure Physical Timer
- Non-Secure Physical Timer
- Hypervisor Timer
- Virtual Timer

Refer to the appropriate ARM Architecture Reference Manual for generic timer specifications.

8.2 Generic Timer System Counter (TSC)

The ARM Generic Timer Specification requires a system time-stamp counter (TSC) supplied by the SoC.

The TSC is implemented as part of the Tegra X1 Power Management Controller (PMC). It is a 56-bit counter which runs at the crystal oscillator clock frequency. During LP0 (when osc may be optionally disabled), the TSC can run with 32,768 kHz clock while still counting at the osc frequency. Refer to Chapter 12: Power Management Controller in this TRM for additional TSC information.

8.3 NVIDIA Timers (TMR)

The programmable timer block contains fourteen 29-bit programmable timer counters (TMRs) and one 32-bit timestamp counter.

The TMRs run at either a fixed 1 MHz clock rate derived from the oscillator clock (TMR0-TMR9]) or directly at the oscillator clock (TMR10-TMR13). Each TMR can be programmed to generate one-shot, periodic, or watchdog interrupts. When a TMR is enabled, it loads the Present Trigger Value (PTV) timer into its counter and starts decrementing at its associated rate. When the Present Count Value (PCR) timer decrements to zero, it generates a timer interrupt. When the periodic interrupt mode is enabled, then timer generates an interrupt and reloads the counter with the PTV value and starts to decrement again.

Each TMR generated (one-shot or periodic) interrupt is routed to an interrupt controller independently so that it can be independently configured to be secure or non-secure. Also, each TMR can be configured to be secure or non-secure (see Section 8.5: Secure TMRs and Secure WDTs).

TMR0 through TMR9 run at a fixed 1 MHz clock rate that is generated from the crystal oscillator clock. However, Tegra X1 devices support several crystal oscillator frequencies. To be able to generate a fixed 1 MHz clock rate, the timer's logic requires "USEC_DIVIDEND" and "USEC_DIVISOR" information to be configured at cold and warm boot (see the TIMERUS_USEC_CFG register description) based on platform oscillator frequency. Note, if the oscillator is disabled, then the timer's logic will not generate the 1 MHz clock. As a result, the Timer and Watchdog will not function when the oscillator is disabled.

The timer count is 29 bits wide, therefore, it can support a periodic interrupt every 536.87 s maximum and 1 µs minimum for TMR0 through TMR9 and platform-dependent limits for TMR10 through TMR13.

8.4 Watchdog Timers (WDTs)

A watchdog timer facilitates recovery from system lockup conditions. The watchdog timer can interrupt the processor when the selected timer counter expires. Under normal operation, this interrupt is serviced by reading the timer status register (TIMER_WDTx_STATUS).

If the interrupt is not serviced by the processor before the watchdog timer counts down to zero for the Nth time (see the TIMER_WDTx_CONFIG register description), this is treated as a lockup condition. When this occurs, the watchdog timer can generate a per processor-reset or system reset, etc.

The Tegra X1 processor provides five Watchdog Timers, one for each pair of CPUs in the fast and slow clusters, and one for BPMP-Lite:

- WDT0 through WDT3 are allocated with CPU<i> of the currently active cluster
- WDT4 is allocated to BPMP-Lite

There are no dedicated timers for the WDTs. Each WDT can be configured to use any one of the TMR0-TMR9 timers as its timer source. Note that TMR10 through TMR13 cannot be selected as source.

Based on the configuration of the TIMER_WDTx_CONFIG register, a WDT works as follows:

- When the timer decrements to zero the first time, a WDT interrupt can be generated
- When the timer consecutively decrements down to zero a second time, without the processor reading the interrupt status register (TIMER_TMRx_TMR_PCR of the corresponding TMR), an FIQ WDT interrupt can be generated. This is not routed directly to the CPU interrupt controller (GIC).
- When the timer consecutively decrements down to zero a third time, without the processor reading the interrupt status register (TIMER_TMRx_TMR_PCR), a per CPU reset can be generated.
- When the timer consecutively decrements down to zero a fourth time without the processor reading the interrupt status register, a system reset can be generated. The cause of this system reset can be read by reading PMC_RST_STATUS[RST_SOURCE] (see Chapter 12: Power Management Controller for more information).

8.5 Secure TMRs and Secure WDTs

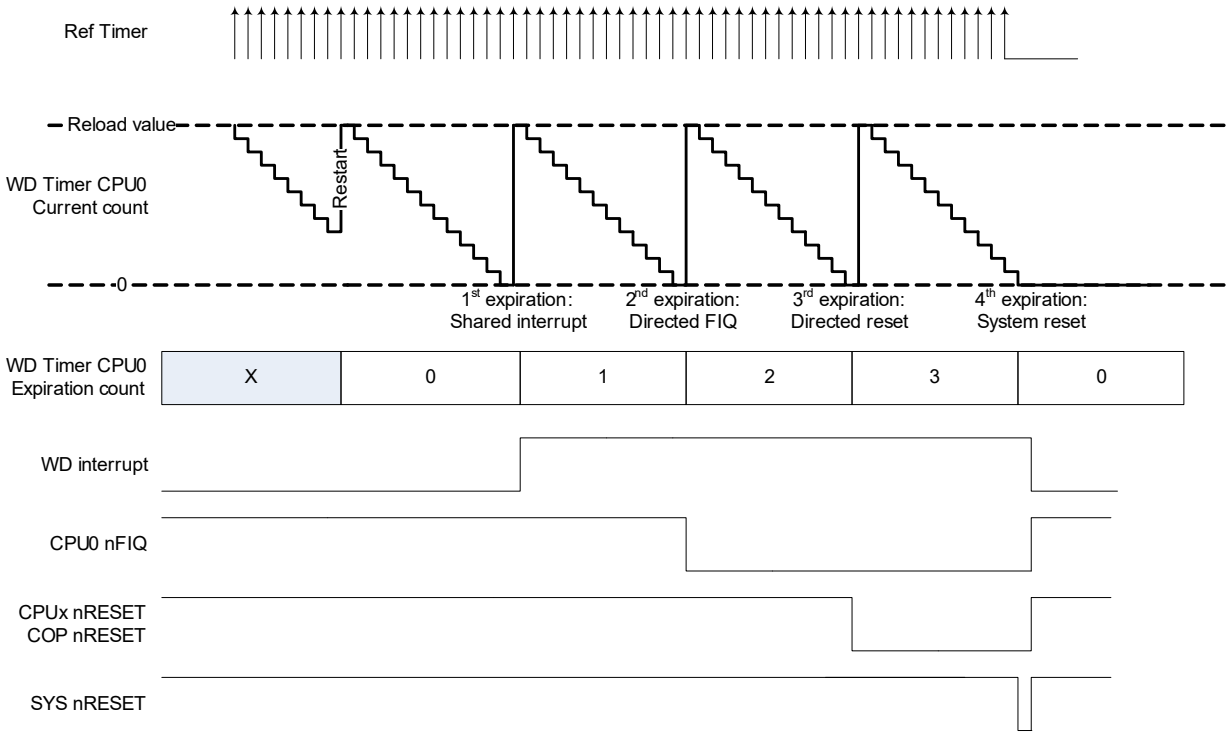
The Tegra X1 processor TMRs and/or WDTs can be configured to be secure timers. A secure register TIMER_SECURE_CFG -- which itself can only be written by secure writes – can be configured such that each of the TMRs and WDTs can be independently configured to be secure. By default all of TMRs and WDTs are non-secure and backward compatible.

8.6 Watchdog Timer Programming Guide

The five watchdog timers operate as follows:

- Select any one of the general-purpose timers as timing reference
- Standard down counter with programmable period, that also counts expirations, specific action are taken at the counter expiration
  - If the expiration count is 0, a normal interrupt may be generated (probably pooled)
  - If the expiration count is 1, the FIQ for the corresponding processor may be asserted, this operation bypasses the normal interrupt controller
  - If the expiration count is 2, the reset for a programmable set of processors is asserted, this may be the null set
  - If the expiration count is 3, a system wide reset may be asserted (two variants, standard system or closer to external hardware reset)

Figure 11: Watchdog Example Timing Diagram



Any operation that targets CPU0 is done in the following way:

- Any output is broadcast to both instances of CPU0, in CPU cluster 0 (G) and 1 (LP). This applies to the reset and FIQ signals.
- Resetting any of the CPU0 results in resetting the WD associated with CPU 0

Although the timing diagram implies that you need to restart the timer when you service the interrupt, this is not necessary as long as you can always service the interrupt (clear the TMR interrupt) before the next timeout occurs. The reset only happens if a new timeout occurs when the TMR interrupt is still active.

8.7 Timers Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Each timer uses two registers called PTV and PCR at two consecutive word addresses. The format is the same for all 14 timers and is defined only once with <t> taking values between 0 and 13. Note that the start offset sequence is haphazard for legacy reasons.

Table 20: Start Offsets for PTV and PCR Timer Registers

Timer	Start Offset for Register Pair
TMR1	0x00
TMR2	0x08
TMR3	0x50
TMR4	0x58
TMR5	0x60
TMR6	0x68
TMR7	0x70
TMR8	0x78
TMR9	0x80
TMR0	0x88
TMR10	0x90
TMR11	0x98
TMR12	0xa0
TMR13	0xa8

8.7.1 TIMER_TMR<t>_TMR_PTV_0

Parameter <t> takes a value from 0 to 13. The start offset for the timer is defined relative to the start address in Table 20.

Before programming the timer:

- Input oscillator frequency must be specified in the TIMERUS_USEC_CFG register.
- Program the number of 1 µs timer counts per “tick” in the PTV (Present Trigger Value) register.
- Set the PTV EN (enable bit) to start counting down.
- When the count reaches zero, an interrupt is generated.
- To auto-reload the timer counter, set the PTV PER (periodic) bit. Otherwise, leave it clear for a one-shot.

Timer Present Trigger Value (Set) Register

Offset: See Table 20 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.2 TIMER_TMR<t>_TMR_PCR_0

Parameter <t> takes a value from 0 to 13. The start offset for the timer is defined relative to the start address in Table 20.

When an interrupt is generated, write “1” to PCR[30] to clear the interrupt

Timer Present Count Value (Status) Register

Offset: See Table 20 + 0x4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no affect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.8 Fixed Time Base Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

The USEC_CFG/CNTR_1US registers provide a fixed time base (in microseconds) to be used by the rest of the system regardless of the clk_m frequency (i.e., 12 MHz or 38.4 MHz).

8.8.1 TIMERUS_CNTR_1US_0

This free-running read-only register/counter changes once every microsecond and is used mainly by hardware (can also be used by software). It starts counting from 0 once it is out of system reset and will continue counting forever, unless the oscillator clock is stopped or during a system reset.

(A) Software Use

Although there is no interrupt mechanism for this register, software can read the content of this register multiple times to determine the amount of time that has elapsed.

(B) Hardware Use

- To provide a periodic USEC pulse to be used by the flow controller to count the programmable number of microseconds before a flow control condition is triggered.
- Used by secure boot logic.

Offset: 0x0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	x	HIGH_VALUE: Elapsed time in microseconds
15:0	x	LOW_VALUE: Elapsed time in microseconds

8.8.2 TIMERUS_USEC_CFG_0

Software should first configure this register by telling what fraction of 1 microsecond each clk_m represents. For example, if the clk_m is running at 12 MHz, then each clk_m represents 1/12 of a microsecond.

"USEC_DIVIDEND" and "USEC_DIVISOR" are used to indicate what fraction of 1 microsecond each clk_m represents.

Refer to Chapter 5: Clock and Reset Controller in this TRM for more information on clk_m.

Table 21: clk_m Frequency Indicator for USEC

clk_m Frequency	Dividend/Divisor	USEC_DIVIDEND/USEC_DIVISOR
12 MHz	1/12	0x00 / 0x0b
38.4 MHz	5/192	0x04 / 0xbf

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxxxxxxxx0000000000001100) | Default: 0x0000000b

Bit	Reset	SW Default	Description
15:8	0x0	NONE	USEC_DIVIDEND: Microsecond dividend. (n+1)

Bit	Reset	SW Default	Description
7:0	0xc	0xb	USEC_DIVISOR: Microsecond divisor. (n+1)

8.8.3 TIMERUS_CNTR_FREEZE_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	DBG_FREEZE_COP: 1 = freeze timers when COP is in debug state, 0 = no freeze.
3	0x0	DBG_FREEZE_CPU3: 1 = freeze timers when CPU3 is in debug state, 0 = no freeze.
2	0x0	DBG_FREEZE_CPU2: 1 = freeze timers when CPU2 is in debug state, 0 = no freeze.
1	0x0	DBG_FREEZE_CPU1: 1 = freeze timers when CPU1 is in debug state, 0 = no freeze.
0	0x0	DBG_FREEZE_CPU0: 1 = freeze timers when CPU0 is in debug state, 0 = no freeze.

8.9 Watchdog Timers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Each Watchdog timer uses four registers called CONFIG, STATUS, COMMAND, and UNLOCK PATTERN, at four consecutive word addresses. The format is the same for all watchdog timers and is defined only once with <w> taking values from 0 to 4.

The following register offsets are relative to the TMR block base.

Each Watchdog timer has the following:

- CoreWatchdog Configuration Register
- Period of 0 results in MAX period
- The generated interrupt is dependent on the WD index.
- WD0 to WD3 generate a CPU_WD_Interrupt
- WD4 generates a COP_WD_Interrupt
- CPU<i> settings apply to both clusters
- Core numbering: 0-3 applies to CPU0-CPU3, 4 applies to COP

8.9.1 TIMER_WDT<w>_CONFIG_0

Parameter <w> takes a value from 0 to 4.

Core Watchdog Configuration Register

Offset: 0x100 + (0x20 * <w>) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at third expiration counter (one bit per processor) same level the signal from PMC asserted in case of external reset.
15	0x0	Pmc2CarResetEn: Enable Full system reset at fourth expiration of the counter 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at fourth expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at second expiration of the counter 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	InterruptEnable: Enable interrupt at first expiration of the counter 0 = DISABLE 1 = ENABLE
11:4	0x0	Period: Measured in periods of selected timer, this is the reload value
3:0	0x0	TimerSource: Select the timer used as reference (TMR0 – TMR9) 0 = TMR0 1 = TMR1 2 = TMR2 3 = TMR3 4 = TMR4 5 = TMR5 6 = TMR6 7 = TMR7 8 = TMR8 9 = TMR9

8.9.2 TIMER_WDT<w>_STATUS_0

Parameter <w> takes a value from 0 to 4.

Core Watchdog Status

Offset: 0x104 + (0x20 * <w>) | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:12	X	CurrentExpirationCount: Current count of expiration since the last start of expiration
11:4	X	CurrentCount: Current value of the counter
2	X	FiqStatus: Current status of FIQ, cleared by start
1	X	InterruptStatus: Current status of interrupt, cleared by start
0	X	Enabled: 1 when counter is active. When true, writes to the corresponding config register are ignored

8.9.3 TIMER_WDT<w>_COMMAND_0

The StartCounter bit enables watchdog counter operation, loads the watchdog counter, starts the watchdog timer to count down, resets the expiration count to 0, and clears all flags. Also used as restart.

The counter can be disabled by setting the DisableCounter bit, but only if the unlock register has been programmed before with the correct pattern. Writing to the command register always clears the disable unlock register. When set while StartCounter is 0 and the unlock register contains the unlock pattern the Watchdog transitions back to disabled.

The register fields are Write to set only.

Parameter <w> takes a value from 0 to 4.

CoreWatchdogCommand Register

Offset: 0x108 + (0x20 * <w>) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

8.9.4 TIMER_WDT<w>_UNLOCK_PATTERN_0

Parameter <w> takes a value from 0 to 4.

CoreWatchdogDisableUnlock Register

Offset: 0x10c + (0x20 * <w>) | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	UnlockPattern: Must be written with value 0xC45A to allow write to DisableCounter to take effect. The pattern is reset at each write to Command Reg.

8.10 Timer Shared Interrupt Status

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

TimerSrcBitmap is used for Timers 6 through 10 which share a common interrupt line to the controller. Bit b is set if timer 6+b generated an interrupt. The corresponding bit is cleared by writing INTR_CLR bit of the TMR_PCR register.

WatchdogSrcBitmap is used for the 5 watchdog timers. This is the set of Interrupt Status bits from the corresponding CoreWatchdogStatus, but as a bitmap.

8.10.1 SHARED_INTR_STATUS_0

Offset: 0x0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:6	X	TimerSrcBitmap: Timer[0,9:6] interrupt status in bitmap form
4:0	X	WatchdogSrcBitmap: WDT[4:0] interrupt status in bitmap form

8.10.2 SHARED_TIMER_SECURE_CFG_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx0xxx00000xx0000000000)

Bit	Reset	Description
27	0x0	TMR13: Timer13 secure mode config. See the TMR0 bit description.
26	0x0	TMR12: Timer12 secure mode config. See the TMR0 bit description.
25	0x0	TMR11: Timer11 secure mode config. See the TMR0 bit description.
24	0x0	TMR10: Timer10 secure mode config. See the TMR0 bit description.
20	0x0	USEC: TIMERUS_USEC_CFG secure mode config. If this bit is set (=1) then TIMERUS_USEC_CFG register can only be written by secure writes. If this bit is cleared then TIMERUS_USEC_CFG register can be written by secure or non-secure writes. This register can always be read by secure or non-secure reads. Note: this bit should be set if any one of the TMR is enabled to be secure. 1: ENABLE 0: DISABLE
16	0x0	WDT4: WDT4 secure mode config. See the WDT0 bit description. This bit is reserved since WDT4 belongs to COP and it has no notion of Secure or Non-Secure transactions
15	0x0	WDT3: WDT3 secure mode config. See the WDT0 bit description.
14	0x0	WDT2: WDT2 secure mode config. See the WDT0 bit description.
13	0x0	WDT1: WDT1 secure mode config. See the WDT0 bit description.
12	0x0	WDT0: WDT0 secure mode config. If this bit is set (=1) then WDT0 registers (i.e. TIMER_WDT0_{CONFIG,COMMAND,UNLOCK_PATTERN}) can only be written by secure writes. A non-secure read will not trigger read-side-effect, but would return proper read value. If this bit is cleared then the WDT0 registers can be written by secure or non-secure writes. WDT0 registers can always be read by secure or non-secure reads. Note: if this bit is set, then the TMR used for this WDT0 should be enabled to be secure. 1: ENABLE 0: DISABLE
9	0x0	TMR9: Timer9 secure mode config. See the TMR0 bit description.
8	0x0	TMR8: Timer8 secure mode config. See the TMR0 bit description.
7	0x0	TMR7: Timer7 secure mode config. See the TMR0 bit description.

Bit	Reset	Description
6	0x0	TMR6: Timer6 secure mode config. See the TMR0 bit description.
5	0x0	TMR5: Timer5 secure mode config. See the TMR0 bit description.
4	0x0	TMR4: Timer4 secure mode config. See the TMR0 bit description.
3	0x0	TMR3: Timer3 secure mode config. See the TMR0 bit description.
2	0x0	TMR2: Timer2 secure mode config. See the TMR0 bit description.
1	0x0	TMR1: Timer1 secure mode config. See the TMR0 bit description.
0	0x0	TMR0: Timer0 secure mode config. If this bit is set (=1) then TMR0 registers (i.e., TIMER_TMR0_TMR_{PTV,PCR}) can only be written by secure writes. A non-secure read will not trigger read-side-effect, but would return proper read value. If this bit is cleared then TMR0 registers can be written by secure or non-secure writes. TMR0 registers can always be read by secure or non-secure reads. 0: DISABLE 1: ENABLE

CHAPTER 9: MULTI-PURPOSE I/O PINS AND PIN MULTIPLEXING (PINMUXING)

9.1 Overview

Tegra® X1 devices can be configured with different I/O functions on particular pins to allow use in a variety of different configurations. This section discusses how this is controlled and how the pins themselves are set up.

Many of the pins on Tegra X1 devices are connected to multi-purpose I/O (MPIO) pads. An MPIO can operate in two modes: either acting as a signal for a particular I/O controller, referred to as a Special-Function I/O (SFIO); or as a software-controlled general-purpose I/O function, referred to as GPIO. Each MPIO has up to four SFIO functions as well as being a GPIO.

Though each MPIO has up to 5 functions (a GPIO function and up to 4 SFIO functions), a given MPIO can only act as a single function at a given point in time. The Pinmux controller in Tegra X1 devices includes the logic and registers to select a particular function for each MPIO.

The VGPIO controller supports 8-bit general-purpose input/output (VGPIO) ports (port A through port D). These ports allow VGPIO signals to be mapped onto unused individual functional pins which might not be present in the same device. This provides a means to virtualize various pins that cannot be mapped in the same package and have to be moved to a companion device.

Note: *The VGPIO controller function is not supported.*

This section describes the following features of MPIOs, the Pinmux controller, the GPIO controller, and the VGPIO controller:

- Basic capabilities of the MPIO pads
- Differences between the variety of MPIO pads
- Mapping of MPIO pads to I/O controllers (i.e., pinmuxing)
- Behavior of the MPIO pads during power-up
- Behavior of the MPIO pads before, during, and after deep sleep
- Recommendations for software programming related to the MPIO pads.

This section covers the Multi-Purpose digital I/O pads. It does not address the special-purpose I/O pads, such as those used for USB, IC_USB, SATA, PCIe, TVO/DAC, MIPI DSI, MIPI CSI, the oscillator, or DRAM interfaces.

9.2 Terms and Acronyms

Some common terms used with Tegra X1 processor pinmuxing are as follows:

Term	Definition
ACPI	Advanced Configuration and Power Interface
CZ	Controlled-output impedance MPIO pads
DD	Dual-driver MPIO pads
Deep sleep	Also known as LP0. A full-chip power state in which VDD_CORE is turned off but VDD_RTC (i.e., VDD_AO) and many of the I/O power rails remain on
DPD	Deep Power Down (a mode in which the pad can tolerate VDD_CORE being turned off)
GPIO	General-Purpose I/O
LP0	See Deep sleep above.

Term	Definition
LV	Low-voltage MPIO pads
MPIO	Multi-Purpose I/O
OD	Open-drain MPIO pads
PMIC	Power Management Integrated Control (synonymous with PMU)
SFIO	Special-Function I/O
ST	Standard MPIO pads
VGPIO	Virtual GPIO

9.3 MPIO Pad Description

Each MPIO pad consists of:

- An output driver with:
  - Tristate capability
  - Drive strength controls AND
  - Push-pull mode, open-drain mode, or both
- An input receiver with:
  - Schmitt mode, CMOS mode, or both
- A weak pull-up and a weak pull-down

Tegra X1 devices include six types of MPIO pads, which all share a common structure. The following table summarizes the differences between the six MPIO pad types.

Table 22: MPIO Pad Types

Pad Type	I/O Rail Voltage (V)	Input Buffer	Output Buffer	I/O Voltage Tolerance	Nominal Pull Strength	"Slew Rate" Control	Drive Strength Control
ST	1.8	Schmitt & CMOS	push-pull	VDDIO	100 kΩ	No	5 bits, up & down
CZ	1.8, 3.3	Schmitt & CMOS	push-pull	VDDIO	18 kΩ (pull-up and pull-down)	2 bits, up & down	7 bits, up & down
DD	1.8	Schmitt & CMOS	push-pull & open-drain	3.3V for open-drain, VDDIO otherwise	100 kΩ	No	5 bits, up & down
LV_CZ	1.2, 1.8	Schmitt & CMOS	push-pull	VDDIO	15 kΩ	2 bits, up & down	5 bits, up & down
ST_EMMC	1.2, 1.8	Schmitt & CMOS	push-pull	VDDIO		3 bits, down only	5 bits, up & down
DP_AUX	1.8	Diff/Open Drain	push-pull & open-drain	3.3V	NA	NA	NA

Table 23: Pad Details and Controls

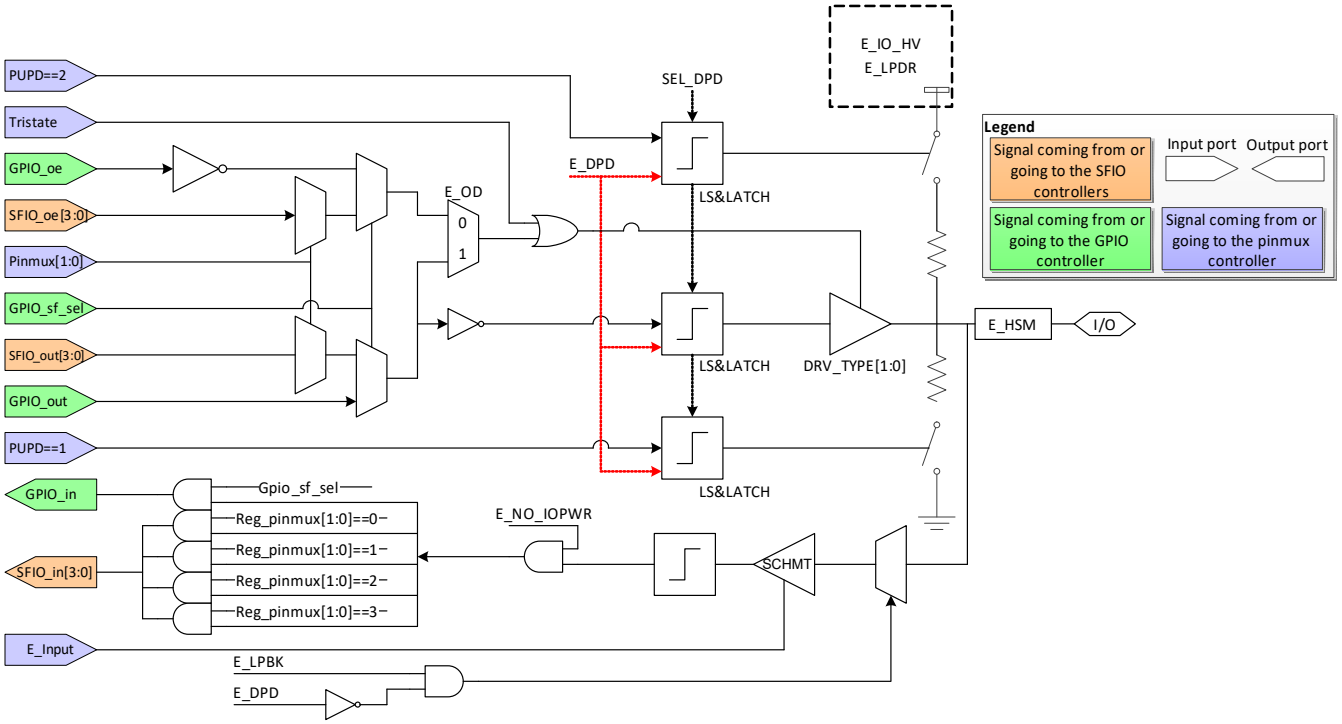
Pad Type Pad Details	ST	CZ	DD	LV_CZ	ST_EMMC
I/O rail voltage	1.8V	1.8V/3.3V	1.8V	1.8V	1.8V
Input Buffer	Schmitt & CMOS	Schmitt & CMOS	Schmitt & CMOS	Schmitt & CMOS	Schmitt & CMOS
Output Buffer	Push-pull	Push-pull	Push-pull & Open Drain	Push-pull	Push-pull
I/O Voltage tolerance	1.8V	VDDIO	Open Drain: 3.3V Otherwise: VDDIO	1.8V	1.8V
Nominal Pull strength	100kΩ	18kΩ (PU and PD)	100kΩ	15kΩ	

Table 23: Pad Details and Controls

Pad Type Pad Details	ST	CZ	DD	LV_CZ	ST_EMMC
"Slew Rate" control	No	2 bits up/down	No	2 bits up/down	3 bits up/down
Drive strength control	5 bits up/down	7 bits up/down	5 bits up/down	5 bits up/down	5 bits up/down
Per Pad Control					
PUPD	√	√	√	√	√
Tristate_control	√	√	√	√	√
DPD_PARKING_CONTROL	√	√	√	√	√
E_INPUT	√	√	√	√	√
E_LPDR	√		√		
E_OD[1]	√		√		
E_IO_HV			√		
E_HSM		√			
SCHMT	√	√	√	√	√
DRV_TYPE[1:0]		√		√	

1. Do not change E_OD. LEAVE IT AT THE DEFAULT VALUE.

Figure 12: Generic MPIO Pad Diagram with Pinmuxing Structure



The ST (standard) MPIO pads are the most common pads on the chip. They are used for typical General Purpose I/O.

The DD (dual-driver) MPIO pads are similar to the ST pads. A DD pad can tolerate its I/O pin being pulled up to 3.3V (regardless of supply voltage) as long as the pad's output-driver is set to open-drain mode. There are special power-sequencing considerations when using this functionality.

> ***Note:*** *Refer to "Power Sequencing" in the NVIDIA Tegra X1 Processors Data Sheet (DS-07224-001) for a complete description of the power-up sequencing requirements for Tegra X1 processors.*

The CZ (controlled output impedance) MPIO pads are optimized for use in applications requiring a tightly controlled output impedance. They are similar to ST pads except for changes in the drive strength circuitry and in the weak pull-ups/pull-downs. Tegra X1 processors include CZ pads on the VDDIO_SDMMC1 and VDDIO_SDDMC3 power rails. Each of those rails also includes a CZ_COMP pad. Because SDMMC3 is connected with a removable SD card, these rails support 3.3V operation. Circuitry within the Tegra X1 device continually matches the output impedance of the CZ pads to the on-board pull-up resistors attached to the CZ_COMP pad. The SDMMC1 interface also can be used for a removable SD card. Apart from this, the Quad SPI interface, which needs to operate around 166 MHz, also uses the BDSDMEM pads to enable 3.3V tolerance.

The LV_CZ (low voltage controlled impedance) pads are similar to CZ pads but are optimized for use with a 1.2V supply voltage (and signaling level). They support a 1.8V supply voltage (and signaling level) as a secondary mode. The Tegra X1 processors use LV_CZ pads for SPI interfaces operating at 1.8V. The eMMC interface uses ST_EMMC, which is the IOBRICK, and supports higher speed DDR operation compliant to eMMC5.0.

The ST_EMMC (Standard and EMMC) pads are I/O bricks that support the eMMC5.0 standard involving DDR and also legacy eMMC standards such as LV_CZ. This way the platforms can connect a new eMMC5.0 based device as well as the legacy devices. These pads support normal ST type of operation so that the bricks can be used as GPIO when it is not used for eMMC. This brick has internal integrated Clock and Calibration pads, used in the SDMMC4 interface which typically connects the secondary boot device. This interface operates in 1.8V mode only because supported eMMC devices operate in 1.8V mode. The same brick is used in the SDMMC2 interface also connects two eMMC5.0 devices to support the high-throughput requirement of Clamshell and Advanced Tablets.

The DP_AUX pad is used as an Auxiliary control channel for the Display Port which needs differential signaling. Because the same I/O brick is used for the Display Port and HDMI to ensure the control path to the display interface is minimized, the DP_AUX pads can operate in open-drain mode so that the HDMI control path (i.e., DDC interface which needs I2C) can also be used in the same pad.

Note: *Refer to "Pin Definitions" in the NVIDIA Tegra X1 Processors Data Sheet (DS-07224-001) for the list of the pad types and the nominal pull-up/-down strength associated with each MPIO. See the columns labeled "MPIO Pad Type" and "Nominal Pull Strength." The power rail information is located in the "Power Sequencing" section.*

9.4 Special Purpose Pads

Beyond the above mentioned pads, there are some special purpose pads used for targeted purposes. These are different from special purpose PHYs used for interfaces such as USB, Display etc. Given below are the special purpose pads.

Table 24: Special Purpose Pad Types

Pad	I/O Rail Voltage	Functional Name	Brief Description
MEM_COMP	1.8 /3.3V	SDMMC Calibration Pads	Used for generating PVT compensated impedance calibration code for the SDMEM pads (typically used for interfacing SDMMC devices). It is connected with an external reference resistor of 50 Ohms with ±1% variation.
JT_RST	1.8V	JTAG and Reset Pads (Input Only)	Used for JTAG (except JTAG TDO) and Reset interface. Has a special mode of operation called VDD only mode where in the pad becomes operational without having the I/O rail up. This mode of operation is needed for Reset pads which have to propagate the reset before I/O rails are up. This is achieved by permanently RCVR_SEL of this pad to Logic 0. For JTAG pads RCVR_SEL is tied to Logic 1 so that it becomes operational when the VDDP rail is up. The TEST_MODE is also made to operate in Core only mode to ensure mode detection for Oscillator which is needed during power on is propagated correctly.

9.5 Pad Controls

The Tegra X1 devices include many controls for each MPIO pad. Some of these controls can be set on a per-pin basis. Other controls are shared across multiple pins.

9.5.1 Per Pad Options

The following controls can be independently configured on a per-pad basis:

Table 25: Per Pad Controls

Control	Description
PUPD	Internal Pull-up/down option: Option to enable internal Pull-up, Pull-down resistors or neither. Applicable to all pads.
TRISTATE_CONTROL	Tristate (high-Z) option: Disables or enables the pad's output driver. This setting overrides any other functional setting and also determines whether the pad is selected for SFIO or GPIO. Can be used when the pad direction changes or the pad is assigned to different SFIO to avoid glitches. During Cold Boot, most of the pads come with this bit set to TRISTATE so that they do not actively drive anything. For Normal Operation, this bit has to be set to PASSTHROUGH state.
DPD_PARKING_CONTROL	The PARKING state holds control during DPD/deep sleep (LP0). During LP0 entry, all pads (except for a few pads in the AO region) are put in the DPD state. This is set in the Pinmux register by default (i.e., as Power on Value). In LP0 exit until this bit is cleared typically by the LP0 exit pinmux recovery code, the pads are in DPD state, i.e., PARKED in the same value as that of LP0 entry. In case the LOCK bit is set for a specific Pinmux register ensure this bit set to Logic1.
E_INPUT	Input Receiver (Enable/Disable): Enables or disables input receiver. Applicable to all pads.
E_LPDR	Enable Base Drivers when set High. Typically set when interfacing chips require minimal rise/fall time such as I2C. Applicable to ST and DD pads.
E_OD	Reserved
E_IO_HV	Enables open-drain pull-up capability to 3.3V thereby enabling High Voltage Operation. Enables 3.3V Receive. If E_IO_HV=1 is set, the pad can support 3.3V open-drain driving with I/O pull-up tolerance up to 3.3V and the Receiver is adjusted to 3.3V DC characteristics. Applicable to DD pads.
E_HSM	High Speed Mode (Enable/Disable). Enables High speed operation for Receiver and Transmitter. Applicable to CZ pads.
SCHMT	Schmitt Trigger (Enable/Disable): Enabling Schmitt provides better noise margin characteristics for the input and, depending on driver's logic threshold levels, this can be enabled. Applicable to all pads
DRV_TYPE[1:0]	Enables different combination of impedance code mapping for the pads. By making this control available to each pad individually, the impedance can be fine-tuned for every pad though the same impedance code is driven to every pad. Applicable to CZ/LV_CZ pads.

During normal operation, these per-pad controls are driven by the pinmux controller registers. See Section 9.6: Pinmuxing below for more information.

During deep sleep, the PMC bypasses and then resets the pinmux controller registers. Software should reprogram these registers as necessary after returning from deep sleep. See Section 9.9: Deep Sleep Behaviors for more information on the interaction of PMC, software, and the pinmux controller following deep sleep.

9.5.2 Per Pad Control Group

The MPIO pads are partitioned into multiple "pad control groups". The following controls can be configured independently for each pad control group. For more flexibility for the I/Os, all the ST and DD pads are given with separate configuration groups for each pads. Though the ST_EMMC pads can be configured as GPIO, there the group of pads has one common control because of how the I/O brick is designed. So the flexibility is not available for balls sharing these I/O bricks.

DRVDN / UP	Drive Down / Up. Driver Output Pull-Up/Pull-Down drive strength code. Normally, the code is 5 bits but for CZ type pads, it is 7 bits.

SLWR/ SLWF	Slew Falling / Rising. Driver Output Pull-Up/Pull-Down slew control code. Normally, the code is 2 bits. In general, the code corresponding to the minimum slew is set (2'b11).

The controls are configured via the "pad control group registers". There is one pad control register per pad control group. During deep sleep, all of these pad control registers automatically return to their power-on-reset state. Software should reprogram these registers as necessary following deep sleep.

Table 26 lists the register address for each pad control group. Additionally, the table describes the bit positions of the controls within each register. Refer to Section 21.1.2: Pad Control Registers in the APB chapter for more details on these registers.

For example, clearing bits 12 through 24 of register 0x7f00008f4 will minimize the drive strength (both up and down) of pads in the AP_MCLK pad control group.

Note: *APB_MISC_GP_SDMMC1_CLK_LPBK_CONTROL_0 (offset 0x8d4) controls the loop back aspect of SDMMC pad. It is defined separately since this control field does not have to be controlled from a functional angle for the other pads.*

Table 26: Pad Control Groups Register Addresses

Pad Control Group	Register Address	SCH	PREEMP	PULLD	PULLU	PARK	DRV_TYPE	DRVDN	DRVUP	SLWR	SLWF
ALS_PROX_INT	0x700008e4							16:12	24:20		
AP_READY	0x700008e8							16:12	24:20		
AP_WAKE_BT	0x700008ec							16:12	24:20		
AP_WAKE_NFC	0x700008f0							16:12	24:20		
AP_MCLK	0x700008f4							16:12	24:20		
BATT_BCL	0x700008f8							16:12	24:20		
BT_RST	0x700008fc							16:12	24:20		
BT_WAKE_AP	0x70000900							16:12	24:20		
BUTTON_HOME	0x70000904							16:12	24:20		
BUTTON_POWER_ON	0x70000908							16:12	24:20		
BUTTON_SLIDE_SW	0x7000090c							16:12	24:20		
BUTTON_VOL_DOWN	0x70000910							16:12	24:20		
BUTTON_VOL_UP	0x70000914							16:12	24:20		
CAM1_MCLK	0x70000918							16:12	24:20		
CAM1_PWDN	0x7000091c							16:12	24:20		
CAM1_STROBE	0x70000920							16:12	24:20		
CAM2_MCLK	0x70000924							16:12	24:20		
CAM2_PWDN	0x70000928							16:12	24:20		
CAM_AF	0x7000092c							16:12	24:20		
CAM_FLASH_EN	0x70000930							16:12	24:20		
CAM_I2C_SCL	0x70000934							16:12	24:20		
CAM_I2C_SDA	0x70000938							16:12	24:20		
CAM_RST	0x7000093c							16:12	24:20		
CLK_32K_IN	0x70000940							16:12	24:20		
CLK_32K_OUT	0x70000944							16:12	24:20		
CLK_REQ	0x70000948							16:12	24:20		
CORE_PWR_REQ	0x7000094c							16:12	24:20		
CPU_PWR_REQ	0x70000950							16:12	24:20		
DAP1_DIN	0x70000954									29:28	31:30
DAP1_DOUT	0x70000958									29:28	31:30
DAP1_FS	0x7000095c									29:28	31:30

Table 26: Pad Control Groups Register Addresses

Pad Control Group	Register Address	SCH	PREEMP	PULLD	PULLU	PARK	DRV_TYPE	DRVDN	DRVUP	SLWR	SLWF
DAP1_SCLK	0x70000960									29:28	31:30
DAP2_DIN	0x70000964									29:28	31:30
DAP2_DOUT	0x70000968									29:28	31:30
DAP2_FS	0x7000096c									29:28	31:30
DAP2_SCLK	0x70000970									29:28	31:30
DAP4_DIN	0x70000974							16:12	24:20		
DAP4_DOUT	0x70000978							16:12	24:20		
DAP4_FS	0x7000097c							16:12	24:20		
DAP4_SCLK	0x70000980							16:12	24:20		
DMIC1_CLK	0x70000984							16:12	24:20		
DMIC1_DAT	0x70000988							16:12	24:20		
DMIC2_CLK	0x7000098c							16:12	24:20		
DMIC2_DAT	0x70000990							16:12	24:20		
DMIC3_CLK	0x70000994							16:12	24:20		
DMIC3_DAT	0x70000998							16:12	24:20		
DP_HPD	0x7000099c							16:12	24:20		
DVFS_CLK	0x700009a0							16:12	24:20		
DVFS_PWM	0x700009a4							16:12	24:20		
GEN1_I2C_SCL	0x700009a8							16:12	24:20		
GEN1_I2C_SDA	0x700009ac							16:12	24:20		
GEN2_I2C_SCL	0x700009b0							16:12	24:20		
GEN2_I2C_SDA	0x700009b4							16:12	24:20		
GEN3_I2C_SCL	0x700009b8							16:12	24:20		
GEN3_I2C_SDA	0x700009bc							16:12	24:20		
GPIO_PA6	0x700009c0							16:12	24:20		
GPIO_PCC7	0x700009c4							16:12	24:20		
GPIO_PE6	0x700009c8							16:12	24:20		
GPIO_PE7	0x700009cc							16:12	24:20		
GPIO_PH6	0x700009d0							16:12	24:20		
GPIO_PK0	0x700009d4									29:28	31:30
GPIO_PK1	0x700009d8									29:28	31:30
GPIO_PK2	0x700009dc									29:28	31:30
GPIO_PK3	0x700009e0									29:28	31:30
GPIO_PK4	0x700009e4									29:28	31:30
GPIO_PK5	0x700009e8									29:28	31:30
GPIO_PK6	0x700009ec									29:28	31:30
GPIO_PK7	0x700009f0									29:28	31:30
GPIO_PL0	0x700009f4									29:28	31:30
GPIO_PL1	0x700009f8									29:28	31:30
GPIO_PZ0	0x700009fc							18:12	26:20		
GPIO_PZ1	0x70000a00							18:12	26:20		
GPIO_PZ2	0x70000a04							18:12	26:20		
GPIO_PZ3	0x70000a08							18:12	26:20		
GPIO_PZ4	0x70000a0c							18:12	26:20		
GPIO_PZ5	0x70000a10							18:12	26:20		
GPIO_X1_AUD	0x70000a14							16:12	24:20		

Table 26: Pad Control Groups Register Addresses

Pad Control Group	Register Address	SCH	PREEMP	PULLD	PULLU	PARK	DRV_TYPE	DRVDN	DRVUP	SLWR	SLWF
GPIO_X3_AUD	0x70000a18							16:12	24:20		
GPS_EN	0x70000a1c							16:12	24:20		
GPS_RST	0x70000a20							16:12	24:20		
HDMI_CEC	0x70000a24							16:12	24:20		
HDMI_INT_DP_HPD	0x70000a28							16:12	24:20		
JTAG_RTCK	0x70000a2c							16:12	24:20		
LCD_BL_EN	0x70000a30							16:12	24:20		
LCD_BL_PWM	0x70000a34							16:12	24:20		
LCD_GPIO1	0x70000a38							16:12	24:20		
LCD_GPIO2	0x70000a3c							16:12	24:20		
LCD_RST	0x70000a40							16:12	24:20		
LCD_TE	0x70000a44							16:12	24:20		
MODEM_WAKE_AP	0x70000a48							16:12	24:20		
MOTION_INT	0x70000a4c							16:12	24:20		
NFC_EN	0x70000a50							16:12	24:20		
NFC_INT	0x70000a54							16:12	24:20		
PEX_L0_CLKREQ_N	0x70000a58							16:12	24:20		
PEX_L0_RST_N	0x70000a5c							16:12	24:20		
PEX_L1_CLKREQ_N	0x70000a60							16:12	24:20		
PEX_L1_RST	0x70000a64							16:12	24:20		
PEX_WAKE_N	0x70000a68							16:12	24:20		
PWR_I2C_SCL	0x70000a6c							16:12	24:20		
PWR_I2C_SDA	0x70000a70							16:12	24:20		
PWR_INT_N	0x70000a74							16:12	24:20		
QSPI_COMP	0x70000a78							18:12	26:20		
QSPI_SCK	0x70000a90									29:28	31:30
SATA_LED_ACTIVE	0x70000a94							16:12	24:20		
SDMMC1_PAD	0x70000a98							18:12	26:20	29:28	31:30
EMMC2_PAD_E	0x70000a9c	0	1			26:14		Note 1	Note 1	29:28	31:30
EMMC2_PAD_DRV_TYPE	0x70000aa0						21:0				
EMMC2_PAD_PUPD	0x70000aa4			Note 2							
SDMMC3_PAD	0x70000ab0							18:12	26:20	29:28	31:30
EMMC4_PAD_E	0x70000ab4	0	1			26:14		Note 1	Note 1	29:28	31:30
EMMC4_PAD_DRV_TYPE	0x70000ab8						21:0				
EMMC4_PAD_PUPD	0x70000abc			Note 2							
SHUTDOWN	0x70000ac8							16:12	24:20		
SPDIF_IN	0x70000acc							16:12	24:20		
SPDIF_OUT	0x70000ad0							16:12	24:20		
SPI1_CS0	0x70000ad4									29:28	31:30
SPI1_CS1	0x70000ad8									29:28	31:30
SPI1_MISO	0x70000adc									29:28	31:30
SPI1_MOSI	0x70000ae0									29:28	31:30
SPI1_SCK	0x70000ae4									29:28	31:30
SPI2_CS0	0x70000ae8									29:28	31:30
SPI2_CS1	0x70000aec									29:28	31:30

Table 26: Pad Control Groups Register Addresses

Pad Control Group	Register Address	SCH	PREEMP	PULLD	PULLU	PARK	DRV_TYPE	DRVDN	DRVUP	SLWR	SLWF
SPI2_MISO	0x70000af0									29:28	31:30
SPI2_MOSI	0x70000af4									29:28	31:30
SPI2_SCK	0x70000af8									29:28	31:30
SPI4_CS0	0x70000afc									29:28	31:30
SPI4_MISO	0x70000b00									29:28	31:30
SPI4_MOSI	0x70000b04									29:28	31:30
SPI4_SCK	0x70000b08									29:28	31:30
TEMP_ALERT	0x70000b0c							16:12	24:20		
TOUCH_CLK	0x70000b10							16:12	24:20		
TOUCH_INT	0x70000b14							16:12	24:20		
TOUCH_RST	0x70000b18							16:12	24:20		
UART1_CTS	0x70000b1c							16:12	24:20		
UART1_RTS	0x70000b20							16:12	24:20		
UART1_RX	0x70000b24							16:12	24:20		
UART1_TX	0x70000b28							16:12	24:20		
UART2_CTS	0x70000b2c							16:12	24:20		
UART2_RTS	0x70000b30							16:12	24:20		
UART2_RX	0x70000b34							16:12	24:20		
UART2_TX	0x70000b38							16:12	24:20		
UART3_CTS	0x70000b3c							16:12	24:20		
UART3_RTS	0x70000b40							16:12	24:20		
UART3_RX	0x70000b44							16:12	24:20		
UART3_TX	0x70000b48							16:12	24:20		
UART4_CTS	0x70000b4c							16:12	24:20		
UART4_RTS	0x70000b50							16:12	24:20		
UART4_RX	0x70000b54							16:12	24:20		
UART4_TX	0x70000b58							16:12	24:20		
USB_VBUS_EN0	0x70000b5c							16:12	24:20		
USB_VBUS_EN1	0x70000b60							16:12	24:20		
WIFI_EN	0x70000b64							16:12	24:20		
WIFI_RST	0x70000b68							16:12	24:20		
WIFI_WAKE_AP	0x70000b6c							16:12	24:20		

Note 1 - These represent the DRVDN and DRVUP codes for the COMP pads.

Note 2 – These represent the PU/PD values for each pin in the brick. 0 = CMD pins PD, 1 = CMD pins PU, 2 = CLK pins PU, etc. Refer to Section 21.1: APB Miscellaneous Registers in the APB chapter for more details.

9.5.3 Per I/O Power Rail

These per-power-rail controls are included in the PMC registers which maintain their state during deep sleep. Software does NOT need to reprogram these register following deep sleep. The following table summarizes the functionality of these signals along with how they are controlled in the PMC.

Table 27: Per Power Rail Controls

Pin Name	Pin Description	Control/Grouping Mechanism in PMC
E_33V	Active high 3.3V mode select. When low, selects VDDP 1.8V mode.	Except for the interface where it is required for boot, it is driven from the register in the PMC which can be configured. By default, it is maintained at Logic 1 to ensure safe power up of I/Os that are pulled to 3.3V

Table 27: Per Power Rail Controls

Pin Name	Pin Description	Control/Grouping Mechanism in PMC
E_NO_IOPWR	Active high. When high, prevents leakage when I/O power is gone while core power is still on.	Maintained per power rails for power rails associated with MPIO. Ideally, should have been set during Cold Reset so that I/O rail voltage ramping does not affect the pad. E_NO_IOPWR is kept at Logic 0. Whenever an I/O power rail is shut off on any specific use case, this bit has to be set to Logic1 before the I/O rails are brought down to avoid excessive leakage to the pad.

9.5.4 Pad DPD Controls

The following signals are individually controlled by the PMC logic as part of power sequencing. These signals help to put the pad in low power states. Typically, most of the pads (except the pads that are part of SYS DPD group) are put into the low power state during the deep sleep (LP0) state of the chip. Apart from this as a use case basis specific set of pads belonging to an individual DPD group(s) can be put in low power state in those cases the pads are not needed for operation.

Table 28: Pad DPD Controls

Pin Name	Pin Description	Control/Grouping Mechanism in PMC
DPD_IO_[1..0]	Programs I/O to Hi/Lo/Tristate for DPD mode. (2 bits)	Controlled via PMC logic by means of latching the I/O value prior to entering the DPD state and driving the same throughout DPD.
SEL_DPD	Selects input source for driving the pad output. SEL_DPD=0 selects the A pin of the pad (which is driven by functional logic) SEL_DPD=1 selects DPD_IO_[1..0].	Generated based on“APBDEV_PMC_DPD_ENABLE_0” in the PMC with some delayed transition so that E_DPD and DPD_IO are at proper values. Deassertion happens based on the removal of DPD_PARKING_CONTROL or DPD_SAMPLE bit (whichever is later). DPD_PARKING_CONTROL is reset after pinmux recovery by the LP0 exit code so that there is no glitch happening during LP0 cycles.
E_DPD	Active high. Places pad in DPD mode by deactivating bias, clamping settings for DPD mode, and gating-out inputs from core	PMC Logic drives during the LP0 cycle Generated based on“APBDEV_PMC_DPD_ENABLE_0” register in PMC”

9.5.5 Special Control for SDMMC CLK Pins

The external loopback feature is controlled by the E_LPBK pin of the SDMMC*CLK pads. Generally, E_LPBK is used by the DFT logic; however, for SDMMC, the functional logic also controls the pin with a default value of Logic 1 to ensure loopback clock is available for SDMMC operation.

Because E_LPBK is set to Logic 1 for the SDMMC*CLK pads when they are used as GPIO, the software has to explicitly ensure the E_LPBK is cleared for correct usage in input mode.

9.6 Pinmuxing

Tegra X1 processors include many types of I/O controllers (such as I^2C, SDMMC, and SPI). For some of these controller types, Tegra X1 processors include multiple “controller instances”. For example, Tegra X1 processors include six active I^2C controller instances.

Each controller instance on a Tegra X1 processor communicates with external devices via a set of “external signals”. For each controller instance, each signal can be brought out on (at least) one MPIO. For example, each I^2C controller has two external signals: CLK and DAT. The CLK signal of the I2C1 controller is available on the MPIO whose ball name is GEN1_I2C_SCL.

Many of the pins on Tegra X1 devices are connected to multi-purpose I/O (MPIO) pads. To connect an MPIO to a particular I/O controller, configure it as a Special-Function I/O (SFIO) rather than a General-Purpose I/O (GPIO). Each MPIO has up to four SFIO functions. Each MPIO can as also be programmed to function as a GPIO. For example, the pin named DMIC1_DAT can act in one of three ways:

- As a GPIO
- As the data signal corresponding to the microphone, i.e., DMIC1_DAT
- As the data signal corresponding to I2S2 interface, i.e., I2S2_SDATA

Though each MPIO supports up to 5 functions (i.e., a GPIO function and up to 4 SFIO functions), a given MPIO can only act as a single function at a given point in time. The Pinmux controller includes the logic and registers to select a particular function for each MPIO.

Some controller instances make their entire set of signals available on two or more sets of MPIOs. That is, such controllers have more than one “interface”. Before using any controller, make sure that the pinmux registers are programmed to bring out the controller’s signals on at most ONE interface.

Note: *The Pinmux controller includes one register per MPIO. It can be controlled on a per-pin basis.*

Figure 12 shows the pinmux logic associated with a single MPIO. The ENB in the diagram (equivalent to the EN pin at the pad) is active Low. Thus, the pad is active when ENB is Low and is in the High-Z state when ENB is High. In the figure, GPIO_OE and SFIO_OE[3:0] are considered active Low; that is, when they are Low, the pad’s output is enabled.

At pad level when ENB =1, the output drivers are disabled. The GPIO unit has different polarity; thus there is an inverter. The polarity for SFIOs is the same as pad level so there is no inverter.

The pinmux controller includes one register per MPIO. The table below describes the layout of each Pinmux controller register.

Table 29: Pinmux Register Format

Field Name	Default Value	Description
LOCK	UNLOCKED	Lock control for writing to the register. Used for security purposes to permanently lock the value to a pinmux register. 0: Writes to this register are accepted 1: Writes to this register are ignored (until the next wake from Deep Sleep) This is a sticky bit. Once software sets this bit to 1, the only way to clear it is to reset the chip or enter and exit Deep Sleep.
E_LPDR	ENABLE/DISABLE (Typically enabled for pads that are used for I2C interface)	Enable only one Base Driver when set High. Typically set when interfacing chips require minimal rise/fall time such as I2C. Applicable to ST and DD pads.
E_OD	DISABLE	DISABLED: Do not use.
E_INPUT	ENABLE	Input Receiver (Enable/Disable): Enables or disables input receiver. Applicable to all the pads.
TRISTATE_CONTROL	TRISTATE/PASSTHROUGH (Most pads have TRISTATE as the default value)	Tristate (High-Z) option: Disables the pad’s output driver. This setting overrides all other functional settings and also whether pad is selected for SFIO or GPIO. Can be used when the pad direction changes or the pad is assigned to a different SFIO to avoid glitches. During Cold Boot, most of the pads come with this bit set to TRISTATE so that they do not actively drive anything. For Normal Operation, the bit has to be set to PASSTHROUGH state. Used by the Pinmux logic to drive the appropriate pad control signals.
PUPD	NORMAL/PU/PD (depending on the pad)	Internal Pull-up/down option: Option to enable internal Pull-up, Pull-down resistors, or neither. Applicable to all the pads.
DPD_PARKING_CONTROL	PARKED (does not affect Cold Boot and comes into effect automatically during LP0 exit)	PARKING state holds control during DPD/LP0. During LP0 (deep sleep) entry, all pads (except a few pads in the AO region) are put in the DPD state This bit is set in the Pinmux register by default during Reset. In LP0 exit until this bit is cleared (typically by the LP0 exit pinmux recovery code), the pads are in the DPD state, i.e., PARKED in the same value as that of LP0 entry. Used by mini_pad_macro logic to appropriately control the DPD controls of the pads. This bit is also clamped to Logic 1 during LP0 entry by the hardware. During LP0 exit until the software clears the bit (the default value being PARKED state) ensures the pads are kept in DPD state. In case the LOCK bit is set for a specific Pinmux register, ensure this bit set to Logic 1 so that deep sleep sequences are proper.

Table 29: Pinmux Register Format

Field Name	Default Value	Description
E_IO_HV	ENABLE/DISABLE (Disabled for PMIC I2C interface)	Enables open-drain pull-up capability to 3.3V, thereby enabling High Voltage Operation. Enables 3.3V Receive. If E_IO_HV=1, the pad can support 3.3V open-drain driving with I/O pull-up tolerance up to 3.3V and the Receiver is adjusted to 3.3V DC characteristics. Default enabled for all the pads so that they can safely operate without knowing whether externally it is pulled up to 3.3V or 1.8V until the pins get used actively. Until that point, it can be driven to High-Z or Logic 0. The platform software can read the status of pull-up values and configure the E_IO_HV before actually using the pin. For the PMIC I2C interface (i.e., PWR_I2C_SCL and PWR_I2C_SDA), it is set at Logic 0 because this interface is needed during boot and the PMIC interface typically has the pull-up at 1.8V. Applicable to DD pads.
E_HSM	DISABLE	High Speed Mode (Enable/Disable). Enables High speed operation for Receiver and Transmitter. Applicable to CZ pads.
SCHMT	DISABLE (for specific interfaces like PMIC the default might be ENABLE depending on the timing requirement)	Schmitt Trigger (Enable/Disable): Enabling Schmitt provides better noise margin characteristics for the input. Depending on driver's logic threshold levels, this can be enabled. Applicable to all pads
DRV_TYPE[1:0]	2X DRIVER MODE (0x01) [This corresponds to 50/33 Ohm impedance configuration]	Enables different combinations of Impedance code mapping for the pads. By making this control available to each pad individually, the impedance can be fine-tuned for every pad though the same impedance code is driven to every pad. Applicable to CZ/LV_CZ pads. For Boot related interfaces like eMMC/QSPI, the default value needed is 0x1 and for the other interfaces it can be changed to 0x3 by the boot loader. For pads in GPIO and AUDIO_HV rails there are no corresponding COMP pads and hence their DRV_TYPE setting should stay in 0x03 only. For SPI_HV rails when they are used for non QSPI purpose the COMP resistor need not have to be mounted and in such cases DRV_TYPE should be set to 0x03 otherwise when COMP resistor is mounted leave to the default value of 0x01.

Refer to the Tegra X1 Pinmux spreadsheet for a list of the SFIO functions supported by each MPIO on Tegra X1 devices. Each SFIO is listed in the table with the following format: <ControllerType><ControllerInstance><InterfaceLetter>-<Signal>

- **ControllerType** indicates the type of I/O controller associated with this SFIO function
- **ControllerInstance** indicates the I/O controller, if the Tegra X1 processor has more than one I/O controller of the specified type
- **InterfaceLetter** differentiates between the pin sets, if the controller instance can connect to more than one set of pins
- **Signal** identifies the particular role this SFIO plays for the I/O controller.

For some of the MPIOs, the hardware includes SFIO functions that are omitted from the spreadsheet. Many of these are deprecated. Contact NVIDIA for more information.

9.7 Cold Boot Reset

The following list is a simplified description of the device power on/boot process for Tegra X1 devices concentrating on those aspects which relate to the MPIO pins.

- System-level hardware executes the power-up sequence. This sequence ends when system-level hardware releases SYS_RESET_N.
- Each MPIO pin has a deterministic power-on-reset state. The particular reset state for each pin is chosen to minimize the need of on-board components like pull-up resistors in a Tegra X1 based system. For example, the on-chip weak pull-ups are enabled during power-on-reset for pins which are usually used to drive active-low chip selects.

- Pins that act as a strap source for cold boot must have their inputs enabled (through E_INPUT control). If they have MPIO Pinmuxing enabled, the strap source should be the primary function.
- The rising edge of SYS_RESET_N does not trigger any of the MPIOs to change their output state.

> ***Note:*** *Refer to "POR Behavior" in the NVIDIA Tegra X1 Processors Data Sheet (DS-07224-001) for a list of the power-on-reset states for each of the MPIOs.*

Following the power-up sequence, most of the MPIOs on the chip will stay in their power-on-reset state until system-dependent software (e.g., the Boot Loader or the OS) reprograms them. However, depending on the secondary boot device used in a given system, the Boot ROM may change the state of some of the MPIOs. The following sections list the MPIOs that the Boot ROM uses for each type of secondary boot device.

9.8 Secondary Boot

The following list is a simplified description of the device boot process for Tegra X1 devices concentrating on those aspects which relate to the MPIO pins.

1. System-level hardware executes the power-up sequence. This sequence ends when system-level hardware releases SYS_RESET_N.
2. The Boot ROM on the Tegra X1 device begins executing and programs the on-chip I/O controllers to access the secondary boot device.
3. The Boot ROM on the Tegra X1 device fetches the BCT and Boot Loader from the secondary boot device.
4. If the BCT and Boot Loader are fetched successfully, Boot ROM on the Tegra X1 device yields to the Boot Loader.
5. Otherwise, Boot ROM on the Tegra X1 device enters USB recovery mode.

Each MPIO pin has a deterministic power-on-reset state. The particular reset state for each pin is chosen to minimize the need of on-board components like pull-up resistors in a Tegra X1 based system. For example, the on-chip weak pull-ups are enabled during power-on-reset for pins which are usually used to drive active-low chip selects.

> ***Note:*** *Refer to "POR Behavior" in the NVIDIA Tegra X1 Processors Data Sheet (DS-07224-001) for a list of the power-on-reset states for each of the MPIOs.*

System designs must adhere to the described power-up sequence.

Following the power-up sequence, most MPIOs on the Tegra X1 device will stay in their power-on-reset state until system-dependent software (the Boot Loader or the OS) reprograms them. However, depending on the secondary boot device used in a given system, the Boot ROM may change the state of some of the MPIOs and associated pinmux so that the Boot ROM can interface with the secondary device to fetch the Boot Loader.

The following subsections list the MPIOs that the Boot ROM uses for each type of secondary boot device. Depending on the type of secondary boot device used, the associated I/O rails have to be brought up by the Boot ROM/PMIC to facilitate the data exchange.

9.8.1 SPI Boot

In systems using a SPI flash as the secondary boot device the Quad SPI controller and associated interface is used. Though it is meant for Quad SPI, it will operate in x1 mode by default and fetch the initial Boot Loader from the SPI device.

All the SPI functionality of Quad SPI is mapped to the primary interface of pinmux. From Boot ROM angle the following needs to be ensured in the respective Pinmux control to enable access to the SPI device:

- Make the TRISTATE_CONTROL to PASSTHROUGH Mode.
- Enable E_INPUT.

The following balls are associated with the Quad SPI and they are part of VDDIO_SPI_HV rail.

Table 30: Pins Associated with SPI Boot

Ball Name	Primary
QSPI_SCK	QSPI_SCK
QSPI_CS_N	QSPI_CS_N
QSPI_IO0	QSPI_IO0
QSPI_ IO1	QSPI_ IO1
QSPI_ IO2	QSPI_ IO2
QSPI_ IO3	QSPI_ IO3

To gain the speed advantage of Quad SPI once the initial Boot Loader is loaded into the Tegra device, it gets executed out of IRAM and the initial Boot Loader has the code for configuring the Quad SPI controller. The following are the broader level steps involved in Boot:

1. [Boot ROM] Initial steps in Boot ROM.
2. [Boot ROM] Fetch BCT, load Initial portion of the Boot Loader to IRAM.
3. [Boot ROM] Jump to IRAM and start executing the Boot Loader.
4. [Boot ROM] Keep the Quad SPI controller interface under idle state.
5. [Boot Loader] Initialize the Quad SPI controller for x4 mode of operation.
6. [Boot Loader] Start the Quad SPI controller and fetch extended Boot Loader, etc.

9.8.2 eMMC Boot

There are four SDMMC interfaces; the one used as a secondary boot interface is SDMMC4. SDMMC4 has all the necessary functionality for SDMMC assigned as primary function, and thus there is no need to change the Pinmux configuration.

From Boot ROM angle the following thing needs to be ensured in the respective Pinmux control to enable access to the SDMMC device

1. Make the TRISTATE_CONTROL to PASSTHROUGH Mode.
2. Enable E_INPUT.

Ball names include:

- SDMMC4_CLK
- SDMMC4_CMD
- SDMMC4_DAT0
- SDMMC4_DAT1
- SDMMC4_DAT2
- SDMMC4_DAT3
- SDMMC4_DAT4
- SDMMC4_DAT5
- SDMMC4_DAT6
- SDMMC4_DAT7
- SDMMC4_COMP

9.8.3 USB RCM Boot

RCM is not a regular boot mode. It is used for recovery purposes (through a combination of key presses). RCM expects that the device is already attached to the host before cold boot, so only D+/D- is used and the other signals are ignored. RCM utilizes

device mode, so current is not provided through VBUS and thus no VBUS condition. Because connection is expected before cold boot, device mode connection detection via valid VBUS is also not used.

9.8.4 SATA Boot

SATA boot requires sideband signals like DEVSLP (Device Sleep-Output) and optionally DA (Data Active-Input) during boot. They require specific pinmux programming considering their mode of operation; for example, Device Sleep operates in Open Drain mode.

9.9 Deep Sleep Behaviors

Deep Sleep is an ultra-low-power standby state in which a Tegra X1 device maintains much of its I/O state while most of the chip is powered off. The following lists offer simplified descriptions of the deep sleep entry and exit concentrating on those aspects which relate to the MPIO pads.

9.9.1 Deep Sleep Cycle

9.9.1.1 Deep Sleep Entry

The steps to enter deep sleep are as follows:

1. LP-entry software programs the PMC to alert about Deep Sleep entry.
2. I/O controllers are brought into the IDLE state.
3. The PMC latches much of Tegra X1's I/O state, i.e., currently driven by functional logic by sampling the pinmux controllers output. There are small variations on this logic depending on the pad type (mentioned below). Latching of the I/O state is triggered by writing into APBDEV_PMC_DPD_SAMPLE_0 register maintained in the PMC register space.
    a. Set the WAKE_DET_EN bit in the APBDEV_PMC_CNTRL2_0 register to Logic 1 while entering deep sleep to ensure the wake status is correctly captured and held correctly for the entire wake.
4. Write the DPD_PARKING_CONTROL bit in the Pinmux register to Logic 1. (This step can be omitted because the default value of DPD_PARKING_CONTROL is in the PARKED state.)
5. Along with latching the I/O values, various pad controls, such as E_IO_HV and E_33V, have to be latched and driven depending on the pad type.
6. The PMC runs the state machine to drive the control pins in the pad that puts them in the DPD state. There are two control pins (E_DPD and SEL_DPD) which have to be driven with proper timing as per the Pad data sheet. The PMC logic provides a configurable timer to select this. The timer is provided in the APBDEV_PMC_SEL_DPD_TIM_0 register. Entering into DPD is triggered through APBDEV_PMC_IO_DPD_REQ_0. Both registers are in the PMC register space.
7. Various clamps are enabled between the SoC to AO domain. Ensure that DPD_PARKING_CONTROL is clamped to Logic1 during deep sleep entry.
8. The PMC deasserts CORE_PWR_REQ.
9. The PMIC powers down VDD_CORE.
10. The PMC continues to drive the I/O state that it has latched.
11. As a result of the deep sleep entry sequence, the pads are put in the DPD state by asserting E_DPD and SEL_DPD. When these signals are asserted, the core supply to the pad (VAUXC_CORE) can be successfully removed and the pad can be powered using VDD_AO/VDD_RTC. This way the pads keep the I/O states yet consume low power.

Note: *For more information about deep sleep entry, refer to the NVIDIA Tegra X1 Processors Data Sheet (DS-07224-001) as well as Chapter 12: Power Management Controller in this TRM.*

9.9.1.2 Deep Sleep Exit

1. The PMC detects a wake event (for example, a rising edge on an appropriately configured MPIO).
2. The PMC asserts CORE_PWR_REQ.
3. The PMIC powers up VDD_CORE.
4. The PMC resets the logic within VDD_CORE.
5. The Boot ROM executes Warm Boot code and does not clear the APBDEV_PMC_DPD_SAMPLE_0 register.
   a. Set the E_INPUT bit of the PINMUX_AUX_GPIO_PA6_0 register to Logic 1.
6. The deep sleep exit code in DRAM retrieves pinmux registers for every pin and ensures all bits are set to their original stored values.
   a. Set the WAKE_DET_EN bit in the APBDEV_PMC_CNTRL2_0 register to Logic 0 to ensure the wake status is correctly captured and held for all the wakes.
7. The PMC takes the I/Os out of E_DPD but continues to drive them in SEL_DPD mode. This way the I/O values do not change during Deep Sleep exit until platform-specific software (i.e., typically customer owned) takes control of the I/Os.
8. The deep sleep -exit code in DRAM re-initializes the I/O controllers.
9. The deep sleep -exit code in DRAM changes the Pinmux configurations to the appropriate controller (based on what was configured during LP0) and clears TRISTATE/PARKED in the Pinmux register of each pin.
10. The deep sleep -exit code in DRAM sets the DPD_PARKING_CONTROL bit in the Pinmux register to Logic 0 in the respective pinmux registers.
11. The deep sleep -exit code in DRAM, after restoring all the Pinmux register, clears the APBDEV_PMC_DPD_SAMPLE_0 register, which results in SEL_DPD removal of the pad through hardware logic.

Note: *For more information about deep sleep exits, refer to the NVIDIA Tegra X1 Processors Data Sheet (DS-07224-001) as well as Chapter 12: Power Management Controller in this TRM.*

9.9.2 Pad Capabilities During Deep Sleep

The MPIO pads do not all have identical behavior during deep sleep. They differ with respect to:

- Output buffer behavior during deep sleep:
  - Does it maintain a programmable (0, 1, or tristate) constant value OR
  - Is it capable of changing state while the chip is still in deep sleep?
- Input buffer behavior during deep sleep:
  - Is it forcibly disabled OR
  - Can it be enabled for use as a “GPIO wake event” OR
  - Can it be enabled for some other purpose (for example, a “clock request” pin)?
- Behavior coming out of deep sleep
  - All the pads should maintain the state until software configures the I/O controller
- Pads that do not enter DPD mode during deep sleep.
  - Some pads whose outputs are dynamic have a special type and they don’t enter DPD mode during deep sleep. The PMC has a configurable option to exclude specific pads from entering DPD during deep sleep (see the PMC register APBDEV_PMC_DPD_PADS_ORIDE_0). Typically, these are pads in VDDIO_SYS domain which are related to system interface like clock, reset, and pins interfacing with mechanical switches
  - Pads that are associated with PMIC logic do not enter DPD mode during deep sleep: CLK_REQ, CPU_PWR_REQ, PWR_INT_N, SHUTDOWN,SYS_RESET_N, CORE_PWR_REQ, CLK_32K_IN

- Some pads like JTAG do not enter DPD mode.

- Special pads in low power state
  - There are two sets of DP_AUX pads corresponding to the HDMI/DP port and eDP/DP port. Both have to be controlled via their Display controller using their internal configuration registers during deep sleep using the appropriate clamps.

Refer to the “Deep Sleep Behavior” section in the *NVIDIA Tegra X1 Processors Data Sheet* (DS-07224-001) for more information on deep sleep.

9.9.2.1 Output Buffers During Deep Sleep

The output buffers of most MPIO pads are “configurable” during deep sleep. Immediately prior to entering deep sleep, the PMC latches the state of the output buffers for such pads. If the pad was driving high (or low) prior to entering deep sleep, the PMC ensures that it continues to drive high (or low) throughout deep sleep. If the pad was tristated during deep sleep, the PMC ensures that it remains tristated throughout deep sleep.

The weak pull-ups and pull-downs of most of the MPIO pads are “configurable” during deep sleep. If the pull-up (or pull-down) was enabled prior to entering deep sleep, the PMC ensures that it remains enabled throughout deep sleep. Similarly, if the pull-up (or pull-down) was disabled prior to entering deep sleep, the PMC ensures that it remains enabled throughout deep sleep.

9.9.2.2 Input Buffers During Deep Sleep

The input buffers of most MPIO pads are “disabled” during deep sleep. The input buffers are placed in an ultra-low power state. The Tegra X1 device will not respond to transitions on these pads while it is in deep sleep.

Some MPIO pads have an input buffer which has a “special” behavior during deep sleep. Most of these pads can act as “Deep Sleep Wake Sources”. Others offer some other functional behavior. For example, SYS_CLK_REQ can function as a clock request pin even during deep sleep.

For a complete description of the behavior of these pads during deep sleep, refer to the *NVIDIA Tegra X1 Processors Data Sheet* (DS-07224-001). Pads whose input can be sampled during Deep Sleep are mentioned as ENABLED in the input buffer column. To ensure the PMC wake logic can sense these wake events, the signals must be directly taken from the ZI pin of the pad because pinmux controls are not available during deep sleep. Pads that are acting as a wake source should satisfy the following condition: The pad’s power rail should be active.

Deep Sleep Debounced Wake

Tegra X1 devices offer debouncing capability to the GPIOs. This enables mechanical push-button switches to be connected to any pins. Most of the mechanical buttons act as deep-sleep wake sources, which means their key press is debounced during deep sleep. However, the debounce logic in GPIO is not in Always ON (AO) domain and hence they are not available during deep sleep. So to enable debouncing of mechanical key press during deep sleep, few balls that act as wake source go through debounce circuit present in PMC during deep sleep to filter out glitches. It is recommended that mechanical switches are connected to these pads.

The following wake pads have these capabilities:

- BUTTON_POWER_ON
- BUTTON_VOL_UP
- BUTTON_VOL_DOWN
- BUTTON_SLIDE_SW
- BUTTON_HOME

9.10 GPIO Controller

The GPIO controller for Tegra X1 devices provides the tools for configuring each MPIO for use as a software-controlled GPIO.

The GPIO controller is divided into 8 banks. Each bank handles the GPIO functionality for up to 32 MPIOs. Within a bank, the GPIOs are arranged as four ports of 8 bits each. The ports are labeled consecutively from A through Z and then AA through FF. Ports A through D are in bank 0. Ports E through H are in bank 1. In total, there are approximately 170 GPIOs, (however, approximately 170 physical GPIOs are available in the chip), and the banking and numbering conventions will have some break in between but will maintain backward compatibility in register configurations for the GPIOs as that of previous generation chips.

Note: *Refer to the "Pin Descriptions" section in NVIDIA Tegra X1 Processors Data Sheet (DS-07224-001) for a map of each of the Tegra X1 device MPIO pads to a particular GPIO port and bit. See the column labeled "GPIO."*

Each GPIO can be individually configurable as Output/Input/Interrupt sources with level/edge controls.

GPIO configuration has a lock bit controlling every bit separately. When the LOCK bit is set, the associated control aspects of the bits (for example, whether it is an Output/Input or used as GPIO or SFIO or values driven) cannot be modified (locked). The LOCK bit gets cleared only by system reset; it is sticky. This bit can be used for security-related functionality where an authorized entity owning the GPIO can set the configuration and lock it. The lock bit also covers the GPIO output value, so this may not be varied dynamically once lock is enabled.

The GPIO controller also has masked-write registers. Values written to these registers specify both a mask of bits to be updated in the underlying state (the mask bits are not sticky) as well as new values for that state. Individual bits of the state can be updated without the need for a read-modify-write sequence. Thus different portions of software can modify the GPIO controller state without coordination.

9.10.1 GPIOs in Loopback Mode

In the specific case of when the GPIO status is polled back because of debugging or other such reasons, the following restrictions arise:

1. If the GPIO output is actively driven the software can immediately poll:
   a. Configure the pin as GPIO output/input.
   b. Drive the output value Active 0 or Active 1.
   c. Poll the input i.e. the same pin or some other pin depending on the loopback configuration.
2. If the GPIO output is configured with the weak internal pull-up or pull-down, then software cannot immediately poll the pin value because the internal pull-up/pull-down have an impedance of around 100K for the pad and the charge/discharge RC delays will govern the time at which the output can be successfully polled. This means the software has to introduce a delay before polling. Conservatively, taking the value of 100K, then 10pF gives RC delays of ~1 µs. The usual required waiting time is 5 RC delays so in the order of 5 µs. To accommodate all the production spread of silicon, software should put a 20 us waiting delay in such cases:
   a. Configure the pin as GPIO output/input.
   b. Bring the output value to weak pull-up or pull-down.
   c. Wait for 20 µs.
   d. Poll the input i.e. the same pin or some other pin depending on the loopback configuration.

9.10.2 Debouncing GPIOs

Tegra X1 devices provide debouncing capabilities for GPIOs. The default is debouncing disabled.

Debouncing logic consists of an Input Synchronizer, Edge Detectors, and Debouncing Counter. Debouncing logic operates off of a 32K clock. The 32K clock is taken as input to the GPIO module.

The debouncing threshold is common to all 8 GPIO pins in a port. The debouncing interval provided is in the order of 128 ms.

The reserved space locations in the existing address space provides debounce control like threshold whether Debounce is enabled or not.

Since the GPIO is not active during deep sleep, the debounce logic is not active during deep sleep. A different debounce logic is used in the PMC for deep sleep wakes that need to be debounced.

9.11 VGPIO Controller

Note: *The VGPIO controller function is not supported.*

The VGPIO controller supports four 8-bit general-purpose input/output ports (port A through port D). These ports allow system designers to map VGPIO signals onto unused individual functional pins which might not be present in the same device. This provides a means to virtualize various pins that cannot be mapped in the same package and have to be moved to a companion device.

The VGPIO_VIN_EN.A – D inputs control the module to select inputs between CPU configured (when in GPIO mode) and system side port status (virtual pin). When this bit is 0, the pin is driven LOW or HIGH according to the state of the OUT bit in the VGPIO_OUT.REG.A – D registers. When this bit is '1', the state is directly taken from a port.

The IN bit of the VGPIO_IN.REG.A – D registers contain the registered input values configured by the ACPI.

9.11.1 VGPIO Features

- Interrupts are programmable on each individual VGPIO pin
- Can act as a GPIO as well as a virtual pin
- Separate interrupt status for input and output directions
- Interrupt levels also programmable
- All pin configurations are independent of the other pins
- Masked-write registers

Each port has a set of eight 8-bit configuration registers, which are used for:

- Receiving the input via ACPI software.
- Enabling interrupts
- Checking status
- Clearing interrupts

In addition, each of these eight registers has a 16-bit version for masked writes.

9.11.2 VGPIO Mapping

Sidebands are available for SD Card Interfaces, and HDMI related pins are moved to Tegra. The USB connection to the controller goes through XUSB_PADCTL which is the wrapper module in Tegra X1 to direct the connection to appropriate USB controller. The XUSB_PADCTL has the appropriate configuration registers to redirect the signal to the correct controller.

Table 31: VGPIO Mapping

Pin Name	Direction	Width	VGPIO Mapping	Tegra X1 Top Level Connection
SDCARD_DETECT1	IN	1	VGPIO.vgpioc_sys_out[0]	Connect directly to appropriate SDMMC controller from either VGPIO or through Tegra GPIOs based on the select control to provide maximum flexibility to the platform.
SDCARD_WP1	IN	1	VGPIO.vgpioc_sys_out[1]	
SDCARD_DETECT2	IN	1	VGPIO.vgpioc_sys_out[2]	
SDCARD_WP2	IN	1	VGPIO.vgpioc_sys_out[3]	

Table 31: VGPIO Mapping

Pin Name	Direction	Width	VGPIO Mapping	Tegra X1 Top Level Connection
HDMI_HPD	IN	1		A direct pin is assigned in Tegra. Note: These indications can be routed to Tegra via primary pin in which case the APB_MISC_GP_VGPIO_GPIO_MUX_SEL_0 should be set to 0.
USB_ID	IN	1	VGPIO.vgpiod_sys_out[0]	Goes from VGPIO through XUSB_PADCTL which appropriately connects to the correct USB controller. Also the USB controller has provided software override to provide more flexibility
USB_ID_A	IN	1	VGPIO.vgpiod_sys_out[1]	
USB_ID_B	IN	1	VGPIO.vgpiod_sys_out[2]	
USB_ID_C	IN	1	VGPIO.vgpiod_sys_out[3]	
USB_OC	IN	1		Provided in Tegra and over loaded with USB_VBUS_EN
USB_B_VLD	IN	1	VGPIO.vgpiod_sys_out[5]	Goes from VGPIO through XUSB_PADCTL which appropriately connects to the correct USB controller. Also the USB controller has provided software override to provide more flexibility
VBUS_WAKEUP	IN	1	VGPIO.vgpiod_sys_out[6]	
VBUS_VLD	IN	1	VGPIO.vgpiod_sys_out[7]	
USB_OTG_PPE (VBUS_ENABLE)	OUT	1		Dedicated Tegra I/Os are provided for VBUS_ENABLE, i.e., USB_VBUS_EN0 & USB_VBUS_EN1
USB2_PPE (VBUS_ENABLE)	OUT	1		
V_GPIO0-7	IN/OUT	8	VGPIO.Port A	Software directly controls it through MMPIO read/write

9.12 Programming Considerations

9.12.1 Pads in Open Drain Mode

Open Drain Pads are used to emulate wired and functionality. The multiple drivers can be shorted on the board with an external pullup. These pads do not drive Active 1, but drives Active 0 or floats the pin. The external onboard pullup takes care of ensuring the correct value on the node (i.e. if all the drivers are floating it is at Logic1; if any of the drivers are driving Active 0, it is at Logic0. Therefore, these pads require a different programming sequence. Any logic (either functional or software based GPIO) should never put pads to Active 1. Both the ST pads and DD pads can operate in Open Drain Mode. But the DD pads have specific control to tolerate the IO swing of 3.3 V.

1. Configure the Pad for 3.3 V swing tolerance by driving the E_IO_HV pin (controlled through Pinmux register) to Logic1.

9.12.2 Controller Instances with Multiple Pin-outs

Several of the I/O controller instances on a Tegra X1 device have more than one pin-out. That is, the controller can communicate with the outside world via more than one set of pins. Special care is warranted when programming the Pinmux controller for these SFIOs. For a given signal on a given controller's interface, only a single pin-out should be selected in the pinmux registers.

The following is the generic procedure to handle multiple pins. For example, assume an output signal from a controller is available on four pins. Depending on the value programmed into the corresponding pinmux registers, each of those four pins might toggle when the controller is sending data. Whichever pin requires this functionality, that pinmux register should be enabled and associate with the controller.

Similarly, if an input signal to the controller is available on four pins then depending on the value programmed into the corresponding pinmux registers, the controller might see the logical-OR of the signal on those four pins. This scenario will almost certainly lead to data corruption. To avoid this, the pinmux registers for the other three pins should be programmed to some other function or disabled through E_INPUT kept at ZERO.

To ensure for controllers whose input can be programmed to sample from multiple pads we must ensure the value driven to the core from the pad is at a constant even if the pads are floating (i.e., unused pin). This is achieved by setting the bit in the

APB_MISC_PP_PINMUX_GLOBAL_0_0 register. Because the default value of TRISTATE/PARKED control in Pinmux is set to Logic 1, it ensures the inputs to the core are clamped at ZERO. The above setting will ensure the input to the core from Pinmux is at Logic 0 irrespective of pinmux configuration. This is as good as setting E_INPUT of the pads to Logic 0 before configuring appropriate pinmux configurations.

9.12.3 Resume From Deep Sleep

As the Tegra X1 device enters Deep Sleep, most of its register state is lost. In particular the pinmux, GPIO, and pad control registers lose their state during deep sleep (LP0). LP0 exit software restores pinmux values, etc. CPU drivers are responsible for reprogramming the I/O controllers, etc. upon resume from deep sleep.

9.12.4 Unused Pins

For each unused MPIO, assert its tristate (TRISTATE_CONTROL) bit and disable its input buffer (E_INPUT) bit. Disable the Pullup/Pull down control to minimize the unnecessary power consumption on the unused pins.

If all of the pins in a pad-control group are unused, set the drive strengths and slew rates to minimum.

If all of the pins on a power-rail are unused, assert E_NOIOPOWER for that rail in the PMC registers.

9.12.5 Security Considerations for the Pinmux Register

Though attacks via Pinmux are mostly Denial Of Service (DoS) attacks, there are specific cases which are a concern, for example:

- Pins interfacing with Platform PMICs (by reprogramming those pinmuxes, one can cause severe user disruption)
- Any I/O device that is transferring secret data (typically like Serial PROMs having keys). By reprogramming the pinmux, it is possible to redirect the data to wrong interface

In such cases, it is better to make use of the LOCK bit in the Pinmux control register so that values cannot be tampered with. Each I/O can be individual locked or unlocked. The Boot Loader can perform this task so that customer software does not have to configure those pins. As part of deep sleep (LP0) exit, the lock bit will have to be re-enabled, because the lock is not retained across deep sleep. Since LP0 exit sequence has multipass iterations, setting the lock bit has to happen on the last pass to ensure all bits are retrieved correctly. Similarly, the GPIO controller has a lock bit for each GPIO to preserve the configuration, value driven, and also the pin is configured as GPIO or SFIO. The configuration can be protected using the lock bit in the GPIO register. This way secure software such as a Boot Loader can ensure that certain pins have the required configuration, irrespective of the actions of less secure software.

9.12.6 Drive Strengths

For all pads that do not have corresponding calibration pads, the drive strength has to be programmed. The common default value assumed for the pad drive impedance is 50 ohms and hence pads have to be programmed with a code that can give 50 ohm impedance. Because the code depends on VDDP, i.e., I/O rail, and also PVT (Process, Voltage and Temperature) corners default a code corresponding to TT-NV-NT (Typical process, Normal Voltage, and Normal Temperature) corner as a function of VDDP voltage is chosen.

In Tegra X1 devices, the primary boot device (SDMMC4) has an auto-calibration option. Thus those pads get an impedance code that matches the PVT conditions, etc. and hence enable the boot process. For all other MPIO pads (which are not involved during boot), the BCT can have a value of the code based on the I/O rail and pad data sheet. The same is programmed by the Boot Loader or any platform-specific software, and it overrides the default Power on Reset values. If the values have to be changed based on SI results, then the same can be loaded in the primary device.

9.13 GPIO Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Each port of each GPIO controller has several registers for the control and monitoring of the port's 8 GPIO pins. The registers provide per-pin control of the following features:

- GPIO_CNF_* / GPIO_MSK_CNF — Select SPIO or GPIO
- GPIO_OE_* / GPIO_MSK_OE_* — Output Enable
- GPIO_OUT_* / GPIO_MSK_OUT_* — GPIO Output Value
- GPIO_IN_* — GPIO Input Value (Read Only)
- GPIO_INT_STA_* / GPIO_MSK_INT_STA_* — GPIO Interrupt Status
- GPIO_INT_ENB_* / GPIO_MSK_INT_ENB_* — Interrupt Enable
- GPIO_INT_LVL_* / GPIO_MSK_INT_LVL_* — Interrupt Selection (Edge/Level)
- GPIO_INT_CLR_* — Interrupt Flag Set-to-Clear
- GPIO_DB_CNT_* — Debounce Counter

Many of the control registers are accessible from two offsets. Accesses to the lower offsets affect all 8 pins for the GPIO port. Accesses to the upper offsets provide a per-pin mask capability allowing adjustments to a subset of the pins. Using the upper offsets can eliminate the need for Read-Modify-Write operations. For example, a write to GPIO_MSK_CNF can substitute for a Read-Modify-Write of GPIO_CNF.

Table 32: GPIO Register Address Map

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO Controller 1 – Port	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
GPIO_CNF_[0/1/2/3]	000	004	008	00C	080	084	088	08C
GPIO_OE_[0/1/2/3]	010	014	018	01C	090	094	098	09C
GPIO_OUT_[0/1/2/3]	020	024	028	02C	0A0	0A4	0A8	0AC
GPIO_IN_[0/1/2/3]	030	034	038	03C				
GPIO_INT_STA_[0/1/2/3]	040	044	048	04C	0C0	0C4	0C8	0CC
GPIO_INT_ENB_[0/1/2/3]	050	054	058	05C	0D0	0D4	0D8	0DC
GPIO_INT_LVL_[0/1/2/3]	060	064	068	06C	0E0	0E4	0E8	0EC
GPIO_INT_CLR_[0/1/2/3]	070	074	078	07C				
The following two registers associated with A,B,C, and D though the Names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					0B0	0B4	0B8	0BC
GPIO_DB_CNT _[P0/P1P2/P3-_0					0F0	0F4	0F8	0FC
GPIO Controller 2 – Port	**E**	**F**	**G**	**H**	**E**	**F**	**G**	**H**
GPIO_CNF_[0/1/2/3]	100	104	108	10C	180	184	188	18C
GPIO_OE_[0/1/2/3]	110	114	118	11C	190	194	198	19C
GPIO_OUT_[0/1/2/3]	120	124	128	12C	1A0	1A4	1A8	1AC
GPIO_IN_[0/1/2/3]	130	134	138	13C				
GPIO_INT_STA_[0/1/2/3]	140	144	148	14C	1C0	1C4	1C8	1CC
GPIO_INT_ENB_[0/1/2/3]	150	154	158	15C	1D0	1D4	1D8	1DC
GPIO_INT_LVL_[0/1/2/3]	160	164	168	16C	1E0	1E4	1E8	1EC
GPIO_INT_CLR_[0/1/2/3]	170	174	178	17C				
The following two registers are associated with E,F,G, and H though the names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					1B0	1B4	1B8	1BC

Table 32: GPIO Register Address Map

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO_DB_CNT _[P0/P1P2/P3-_0					1F0	1F4	1F8	1FC
GPIO Controller 3 – Port	**I**	**J**	**K**	**L**	**I**	**J**	**K**	**L**
GPIO_CNF_[0/1/2/3]	200	204	208	20C	280	284	288	28C
GPIO_OE_[0/1/2/3]	210	214	218	21C	290	294	298	29C
GPIO_OUT_[0/1/2/3]	220	224	228	22C	2A0	2A4	2A8	2AC
GPIO_IN_[0/1/2/3]	230	234	238	23C				
GPIO_INT_STA_[0/1/2/3]	240	244	248	24C	2C0	2C4	2C8	2CC
GPIO_INT_ENB_[0/1/2/3]	250	254	258	25C	2D0	2D4	2D8	2DC
GPIO_INT_LVL_[0/1/2/3]	260	264	268	26C	2E0	2E4	2E8	2EC
GPIO_INT_CLR_[0/1/2/3]	270	274	278	27C				
The following two registers associated with I,J,K, and L though the Names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					2B0	2B4	2B8	2BC
GPIO_DB_CNT _[P0/P1P2/P3-_0					2F0	2F4	2F8	2FC
GPIO_CNF_[0/1/2/3]	200	204	208	20C	280	284	288	28C
GPIO_OE_[0/1/2/3]	210	214	218	21C	290	294	298	29C
GPIO Controller 4 – Port	**M**	**N**	**O**	**P**	**M**	**N**	**O**	**P**
GPIO_CNF_[0/1/2/3]	300	304	308	30C	380	384	388	38C
GPIO_OE_[0/1/2/3]	310	314	318	31C	390	394	398	39C
GPIO_OUT_[0/1/2/3]	320	324	328	32C	3A0	3A4	3A8	3AC
GPIO_IN_[0/1/2/3]	330	334	338	33C				
GPIO_INT_STA_[0/1/2/3]	340	344	348	34C	3C0	3C4	3C8	3CC
GPIO_INT_ENB_[0/1/2/3]	350	354	358	35C	3D0	3D4	3D8	3DC
GPIO_INT_LVL_[0/1/2/3]	360	364	368	36C	3E0	3E4	3E8	3EC
GPIO_INT_CLR_[0/1/2/3]	370	374	378	37C				
The following two registers associated with M,N,O, and P though the Names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					3B0	3B4	3B8	3BC
GPIO_DB_CNT _[P0/P1P2/P3-_0					3F0	3F4	3F8	3FC
GPIO Controller 5 – Port	**Q**	**R**	**S**	**T**	**Q**	**R**	**S**	**T**
GPIO_CNF_[0/1/2/3]	400	404	408	40C	480	484	488	48C
GPIO_OE_[0/1/2/3]	410	414	418	41C	490	494	498	49C
GPIO_OUT_[0/1/2/3]	420	424	428	42C	4A0	4A4	4A8	4AC
GPIO_IN_[0/1/2/3]	430	434	438	43C				
GPIO_INT_STA_[0/1/2/3]	440	444	448	44C	4C0	4C4	4C8	4CC
GPIO_INT_ENB_[0/1/2/3]	450	454	458	45C	4D0	4D4	4D8	4DC
GPIO_INT_LVL_[0/1/2/3]	460	464	468	46C	4E0	4E4	4E8	4EC
GPIO_INT_CLR_[0/1/2/3]	470	474	478	47C				
The following two registers associated with Q,R,S, and T though the Names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					4B0	4B4	4B8	4BC

Table 32: GPIO Register Address Map

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO_DB_CNT _[P0/P1P2/P3-_0					4F0	4F4	4F8	4FC
GPIO Controller 6 – Port	**U**	**V**	**W**	**X**	**U**	**V**	**W**	**X**
GPIO_CNF_[0/1/2/3]	500	504	508	50C	580	584	588	58C
GPIO_OE_[0/1/2/3]	510	514	518	51C	590	594	598	59C
GPIO_OUT_[0/1/2/3]	520	524	528	52C	5A0	5A4	5A8	5AC
GPIO_IN_[0/1/2/3]	530	534	538	53C				
GPIO_INT_STA_[0/1/2/3]	540	544	548	54C	5C0	5C4	5C8	5CC
GPIO_INT_ENB_[0/1/2/3]	550	554	558	55C	5D0	5D4	5D8	5DC
GPIO_INT_LVL_[0/1/2/3]	560	564	568	56C	5E0	5E4	5E8	5EC
GPIO_INT_CLR_[0/1/2/3]	570	574	578	57C				
The following two registers associated with U,V,W, and X though the Names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					5B0	5B4	5B8	5BC
GPIO_DB_CNT _[P0/P1P2/P3-_0					5F0	5F4	5F8	5FC
GPIO Controller 7 – Port	**Y**	**Z**	**AA**	**BB**	**Y**	**Z**	**AA**	**BB**
GPIO_CNF_[0/1/2/3]	600	604	608	60C	680	684	688	68C
GPIO_OE_[0/1/2/3]	610	614	618	61C	690	694	698	69C
GPIO_OUT_[0/1/2/3]	620	624	628	62C	6A0	6A4	6A8	6AC
GPIO_IN_[0/1/2/3]	630	634	638	63C				
GPIO_INT_STA_[0/1/2/3]	640	644	648	64C	6C0	6C4	6C8	6CC
GPIO_INT_ENB_[0/1/2/3]	650	654	658	65C	6D0	6D4	6D8	6DC
GPIO_INT_LVL_[0/1/2/3]	660	664	668	66C	6E0	6E4	6E8	6EC
GPIO_INT_CLR_[0/1/2/3]	670	674	678	67C				
The following two registers associated with Y,Z,AA, and BB though the Names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					6B0	6B4	6B8	6BC
GPIO_DB_CNT _[P0/P1P2/P3-_0					6F0	6F4	6F8	6FC
GPIO Controller 8 – Port	**CC**	**DD**	**EE**		**CC**	**DD**	**EE**	
GPIO_CNF_[0/1/2/3]	700	704	708		780	784	788	
GPIO_OE_[0/1/2/3]	710	714	718		790	794	798	
GPIO_OUT_[0/1/2/3]	720	724	728		7A0	7A4	7A8	
GPIO_IN_[0/1/2/3]	730	734	738					
GPIO_INT_STA_[0/1/2/3]	740	744	748		7C0	7C4	7C8	
GPIO_INT_ENB_[0/1/2/3]	750	754	758		7D0	7D4	7D8	
GPIO_INT_LVL_[0/1/2/3]	760	764	768		7E0	7E4	7E8	
GPIO_INT_CLR_[0/1/2/3]	770	774	778					
The following two registers associated with CC,DD, EE, and FF though the Names are mentioned as P0/P1/P2/P3. They are positioned at the Upper address holes of GPIO_IN_ and GPIO_INT_CLR to contain the address range.								
GPIO_DB_CTRL_[P0/P1/P2/P3]_0					7B0	7B4	7B8	7BC
GPIO_DB_CNT _[P0/P1P2/P3-_0					7F0	7F4	7F8	7FC
GPIO_CNF_[0/1/2/3]	700	704	708		780	784	788	

Table 32: GPIO Register Address Map

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO_OE_[0/1/2/3]	710	714	718		790	794	798	

In Tegra X1 devices, E_LPBK for SDMMC1_CLK and SDMMC3_CLK pads have a default value of 1. Enabling E_LPBK makes Zi = A (bypassing the I/O). To use this pad as INPUT, E_LPBK has to be programmed to 0. This can be achieved by programming the following registers:

- APB_MISC_GP_SDMMC1_CLK_LPBK_CONTROL_0 for sdmmc1_clk pad
- APB_MISC_GP_SDMMC3_CLK_LPBK_CONTROL_0 for sdmmc3_clk pad

Note:

- SDMMC1_CLK pad is connected to pin '0' of port M in GPIO
- SDMMC3_CLK pad is connected to pin '0' of port P in GPIO

9.13.1 GPIO_CNF_0

Designates whether each pin operates as a GPIO or as an SFIO. By default all pins come up in SFIO mode. These can be programmed to GPIO mode at any stage.

Lock bits are used to control the access to the CNF and OE registers. When set, no one can write to the CNF and OE bits. They can be programmed ONLY during Boot and get reset by chip reset only.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port A - D Configuration Registers

Offset: 0h..fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15	DISABLE	LOCK_7: Lock access to pin 7 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
14	DISABLE	LOCK_6: Lock access to pin 6 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
13	DISABLE	LOCK_5: Lock access to pin 5 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
12	DISABLE	LOCK_4: Lock access to pin 4 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
11	DISABLE	LOCK_3: Lock access to pin 3 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
10	DISABLE	LOCK_2: Lock access to pin 2 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
9	DISABLE	LOCK_1: Lock access to pin 1 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
8	DISABLE	LOCK_0: Lock access to pin 0 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
7	0x0	BIT_7: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
6	0x0	BIT_6: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO

Bit	Reset	Description
5	0x0	BIT_5: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
4	0x0	BIT_4: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
3	0x0	BIT_3: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
2	0x0	BIT_2: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
1	0x0	BIT_1: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
0	0x0	BIT_0: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO

9.13.2 GPIO_OE_0

GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid.

The set of registers below are used to either drive the signal out or as an Input. This needs to be programmed depending upon whether the pin needs to be in either Input or Output.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Enable Registers

Offset: 10h..1fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = TRI_STATE 1 = DRIVEN
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN

9.13.3 GPIO_OUT_0

GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be valid. This register will take affect only in GPIO mode. This register is used to drive the value out on a given pin.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Output Value Registers (Read/Write)

Offset: 20h..2fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
6	0x0	BIT_6: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
5	0x0	BIT_5: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
4	0x0	BIT_4: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
3	0x0	BIT_3: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
2	0x0	BIT_2: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
1	0x0	BIT_1: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
0	0x0	BIT_0: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH

9.13.4 GPIO_IN_0

GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. This is a read-only register used to read the value from the pin. This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Input Value Registers (Read Only)

Offset: 30h..3fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = LOW 1 = HIGH
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH

Bit	Reset	Description
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH

All GPIO inputs can be independently programmed to generate an interrupt request.

In addition, the individual trigger level for interrupt on each input pin can be programmed as either active-on-high or active-on-low. For example, to program an active-on-high interrupt on bit 3 of GPIO-PORT_C, write '1' into bit 3 of GPIO_INT.LVL.C register (this sets the interrupt to be active-on-high), and then write '1' into bit 3 of GPIO_INT.ENB.C (this enables interrupt on the named bit).

The interrupt flag status can be read in the appropriate bit of the GPIO_INT.STA.C register. Once the programmed interrupt occurs, status should be cleared by writing into the appropriate bit of the GPIO_INT.CLR.C register. Note that the interrupt thus generated is routed to the processor only if the corresponding bit for GPIO interrupts in the Secondary interrupt controller is enabled.

9.13.5 GPIO_INT_STA_0

GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. Every GPIO pin generates an Interrupt when switching from Low-High to High-Low. Interrupt status for each port is saved in an Interrupt status register.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Status Registers

Offset: 40h..4fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE

Bit	Reset	Description
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE

9.13.6 GPIO_INT_ENB_0

Every bit of the GPIO pin has an enable which, when enabled, routes the Interrupt to the Interrupt controller. This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Enable Registers

Offset: 50h..5fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE

9.13.7 GPIO_INT_LVL_0

The GPIO can detect an interrupt for any edge- or level-sensitive signal.

This is an array of 4 identical register entries; the register fields below apply to each entry

GPIO Port Interrupt Activation Level Registers

Offset: 60h..6fh | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23	0x0	DELTA_7: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
22	0x0	DELTA_6: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
21	0x0	DELTA_5: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
20	0x0	DELTA_4: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
19	0x0	DELTA_3: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
18	0x0	DELTA_2: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
17	0x0	DELTA_1: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
16	0x0	DELTA_0: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
15	0x0	EDGE_7: 1 means Configure as Edge-Triggered Interrupt
14	0x0	EDGE_6: 1 means Configure as Edge-Triggered Interrupt
13	0x0	EDGE_5: 1 means Configure as Edge-Triggered Interrupt
12	0x0	EDGE_4: 1 means Configure as Edge-Triggered Interrupt
11	0x0	EDGE_3: 1 means Configure as Edge-Triggered Interrupt
10	0x0	EDGE_2: 1 means Configure as Edge-Triggered Interrupt
9	0x0	EDGE_1: 1 means Configure as Edge-Triggered Interrupt
8	0x0	EDGE_0: 1 means Configure as Edge-Triggered Interrupt
7	0x0	BIT_7: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
6	0x0	BIT_6: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
5	0x0	BIT_5: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
4	0x0	BIT_4: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
3	0x0	BIT_3: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
2	0x0	BIT_2: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
1	0x0	BIT_1: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
0	0x0	BIT_0: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH

9.13.8 GPIO_INT_CLR_0

This write-only register clears the Interrupts that are set. This is valid only in GPIO mode when GPIO_INT.ENB is set.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Flag Set-to-Clear Registers

Offset: 70h..7fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR

9.13.9 GPIO_MSK_CNF_0

Each register is provided with an individual 16-bit version for enabling Masked Writes to avoid a Read-Modify-Write operation by the firmware. The exception is for the interrupt clear register, whose functionality is combined in the interrupt status register. Individual pins only can be programmed by suitably enabling the write masks in the upper byte of these 16-bit registers.

This is an array of 4 identical register entries; the register fields below apply to each entry.

MASKED PRIMARY GPIO/SFIO Config Registers (Masked Writes)

Offset: 80h..8fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = SPIO 1 = GPIO
6	RW	0x0	BIT_6: 0 = SPIO 1= GPIO
5	RW	0x0	BIT_5: 0 = SPIO 1 = GPIO
4	RW	0x0	BIT_4: 0 = SPIO 1 = GPIO
3	RW	0x0	BIT_3: 0 = SPIO 1 = GPIO
2	RW	0x0	BIT_2: 0 = SPIO 1 = GPIO
1	RW	0x0	BIT_1: 0 = SPIO 1 = GPIO
0	RW	0x0	BIT_0: 0 = SPIO 1 = GPIO

9.13.10 GPIO_MSK_OE_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Masked Output Enable (Masked Writes)

Offset: 90h..9fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = TRI_STATE 1 = DRIVEN
6	RW	0x0	BIT_6: 0 = TRI_STATE 1 = DRIVEN
5	RW	0x0	BIT_5: 0 = TRI_STATE 1 = DRIVEN
4	RW	0x0	BIT_4: 0 = TRI_STATE 1 = DRIVEN
3	RW	0x0	BIT_3: 0 = TRI_STATE 1 = DRIVEN
2	RW	0x0	BIT_2: 0 = TRI_STATE 1 = DRIVEN
1	RW	0x0	BIT_1: 0 = TRI_STATE 1 = DRIVEN
0	RW	0x0	BIT_0: 0 = TRI_STATE 1 = DRIVEN

9.13.11 GPIO_MSK_OUT_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Output Enable (Masked Writes)

Offset: a0h..afh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

9.13.12 GPIO_DB_CTRL_P0_0

GPIO PORT 0 Debounce Control Register

P0_DBC_EN: Debounce enable.

- Bit 7: Enables debounce logic for Port 0 pin 7
- Bit 6: Enables debounce logic for Port 0 pin 6
- Bit 5: Enables debounce logic for Port 0 pin 5
- Bit 4: Enables debounce logic for Port 0 pin 4
- Bit 3: Enables debounce logic for Port 0 pin 3
- Bit 2: Enables debounce logic for Port 0 pin 2
- Bit 1: Enables debounce logic for Port 0 pin 1
- Bit 0: Enables debounce logic for Port 0 pin 0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15:8	WO	0x0	MSK_P0_DBC_EN: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7:0	RW	0x0	P0_DBC_EN: 0 = LOW 1 = HIGH

9.13.13 GPIO_DB_CTRL_P1_0

GPJO PORT 1 Debounce Control Register

P1_DBC_EN: Debounce enable.

- Bit 7: Enables debounce logic for Port 1 pin 7
- Bit 6: Enables debounce logic for Port 1 pin 6
- Bit 5: Enables debounce logic for Port 1 pin 5
- Bit 4: Enables debounce logic for Port 1 pin 4
- Bit 3: Enables debounce logic for Port 1 pin 3
- Bit 2: Enables debounce logic for Port 1 pin 2
- Bit 1: Enables debounce logic for Port 1 pin 1
- Bit 0: Enables debounce logic for Port 1 pin 0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15:8	WO	0x0	MSK_P1_DBC_EN: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7:0	RW	0x0	P1_DBC_EN: 0 = LOW 1 = HIGH

9.13.14 GPIO_DB_CTRL_P2_0

GPKO PORT 2 Debounce Control Register

P2_DBC_EN: Debounce enable.

- Bit 7: Enables debounce logic for Port 2 pin 7
- Bit 6: Enables debounce logic for Port 2 pin 6
- Bit 5: Enables debounce logic for Port 2 pin 5
- Bit 4: Enables debounce logic for Port 2 pin 4
- Bit 3: Enables debounce logic for Port 2 pin 3
- Bit 2: Enables debounce logic for Port 2 pin 2
- Bit 1: Enables debounce logic for Port 2 pin 1
- Bit 0: Enables debounce logic for Port 2 pin 0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15:8	WO	0x0	MSK_P2_DBC_EN: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7:0	RW	0x0	P2_DBC_EN: 0 = LOW 1 = HIGH

9.13.15 GPIO_DB_CTRL_P3_0

GPLO PORT 3 Debounce Control Register

P3_DBC_EN: Debounce enable.

- Bit 7: Enables debounce logic for Port 3 pin 7
- Bit 6: Enables debounce logic for Port 3 pin 6
- Bit 5: Enables debounce logic for Port 3 pin 5
- Bit 4: Enables debounce logic for Port 3 pin 4
- Bit 3: Enables debounce logic for Port 3 pin 3
- Bit 2: Enables debounce logic for Port 3 pin 2
- Bit 1: Enables debounce logic for Port 3 pin 1
- Bit 0: Enables debounce logic for Port 3 pin 0

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15:8	WO	0x0	MSK_P3_DBC_EN: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7:0	RW	0x0	P3_DBC_EN: 0 = LOW 1 = HIGH

9.13.16 GPIO_MSK_INT_STA_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Interrupt Status (Masked Clears)

Offset: c0h..cfh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = IN_ACTIVE 1 = ACTIVE
6	RW	0x0	BIT_6: 0 = IN_ACTIVE 1 = ACTIVE
5	RW	0x0	BIT_5: 0 = IN_ACTIVE 1 = ACTIVE
4	RW	0x0	BIT_4: 0 = IN_ACTIVE 1 = ACTIVE
3	RW	0x0	BIT_3: 0 = IN_ACTIVE 1 = ACTIVE
2	RW	0x0	BIT_2: 0 = IN_ACTIVE 1 = ACTIVE
1	RW	0x0	BIT_1: 0 = IN_ACTIVE 1 = ACTIVE
0	RW	0x0	BIT_0: 0 = IN_ACTIVE 1 = ACTIVE

9.13.17 GPIO_MSK_INT_ENB_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Interrupt Enable (Masked Writes)

Offset: d0h..dfh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = DISABLE 1 = ENABLE
6	RW	0x0	BIT_6: 0 = DISABLE 1 = ENABLE
5	RW	0x0	BIT_5: 0 = DISABLE 1 = ENABLE
4	RW	0x0	BIT_4: 0 = DISABLE 1 = ENABLE
3	RW	0x0	BIT_3: 0 = DISABLE 1 = ENABLE
2	RW	0x0	BIT_2: 0 = DISABLE 1 = ENABLE
1	RW	0x0	BIT_1: 0 = DISABLE 1 = ENABLE
0	RW	0x0	BIT_0: 0 = DISABLE 1 = ENABLE

9.13.18 GPIO_MSK_INT_LVL_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Masked Write Interrupt Activation Levels

Offset: e0h..efh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

9.13.19 GPIO_DB_CNT_P0_0

GPIO PORT 0 Debounce Control Register

Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	P0_DBC_CNT: Debounce count. This value sets the debounce count with respect to 1ms period. The interval is associated with the input transitions to capture the input after the specific Debouncing period is over. 0 = No debounce N = N ms

9.13.20 GPIO_DB_CNT_P1_0

GPJO PORT 1 Debounce Control Register

Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	P1_DBC_CNT: Debounce count. This value sets the debounce count with respect to 1ms period. The interval is associated with the input transitions to capture the input after the specific Debouncing period is over. 0 = No debounce N = N ms

9.13.21 GPIO_DB_CNT_P2_0

GPKO PORT K Debounce Control Register

Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	P2_DBC_CNT: Debounce count. This value sets the debounce count with respect to 1ms period. The interval is associated with the input transitions to capture the input after the specific Debouncing period is over. 0 = No debounce N = N ms

9.13.22 GPIO_DB_CNT_P3_0

GPLO PORT 3 Debounce Control Register

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	P3_DBC_CNT: Debounce count. This value sets the debounce count with respect to 1ms period. The interval is associated with the input transitions to capture the input after the specific Debouncing period is over. 0 = No debounce N = N ms

9.14 VGPIO Registers

Each VGPIO Controller has four 8-bit VGPIO interfaces. Each pin can be programmed independently.

A VGPIO interrupt can be enabled, status checked, and cleared for any of these pins individually, the interrupt trigger level being programmable. Each port has a set of eight 8-bit registers, for enabling output, receiving the input, and finally, for enabling the interrupt, checking the status, and clearing the interrupt.

Each of these eight registers has a 16-bit version for masked writes to avoid doing a Read-Modify-Write. The interrupt clear register is an exception to this; its functionality is combined in the interrupt status register.

Each port of each VGPIO controller has several registers for the control and monitoring of the port's 8 VGPIO pins. The registers provide per-pin control of the following features:

- VGPIO_LCK — Lock bits for OE
- VGPIO_OE_* / GPIO_MSK_OE_* — Output Enable
- VGPIO_OUT_* / GPIO_MSK_OUT_* — VGPIO Output Value
- VGPIO_IN_*/ GPIO_MSK_IN_* — VGPIO Input Value (Read/Write)
- VGPIO_INT_STA_* /VGPIO_MSK_INT_STA_* — VGPIO Interrupt Status
- VGPIO_INT_ENB_* /VGPIO_MSK_INT_ENB_* — Interrupt Enable

- VGPIO_INT_LVL_* Interrupt Selection (Edge/Level)
- VGPIO_INT_CLR_* Interrupt Flag Set-to-Clear
- VGPIO_INT_STA_ACPI_*/VGPIO_MSK_INT_STA_ACPI* V-GPIO ACPI Interrupt Status
- VGPIO_INT_CLR_ACPI_* ACPI Interrupt Flag Set-to-Clear

Many of the control registers are accessible from two offsets. Accesses to the lower offsets affect all 8 pins for the VGPIO port. Accesses to the upper offsets provide a per-pin mask capability allowing adjustments to a subset of the pins. Using the upper offsets can eliminate the need for Read-Modify-Write operations. For example, a write to VGPIO_MSK_OUT can substitute for a Read-Modify-Write of VGPIO_OUT.

Table 33: VGPIO Register Address Map

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO Controller 1 – Port	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
VGPIO_LCK	000	004	008	00C				
VGPIO_OE	010	014	018	01C	090	094	098	09C
VGPIO_OUT	020	024	028	02C	0A0	0A4	0A8	0AC
VGPIO_IN	030	034	038	03C	0B0	0B4	0B8	0BC
VGPIO_INT_STA	040	044	048	04C	0C0	0C4	0C8	0CC
VGPIO_INT_ENB	050	054	058	05C	0D0	0D4	0D8	0DC
VGPIO_INT_LVL	060	064	068	06C				
VGPIO_INT_CLR	070	074	078	07C				
VGPIO_INT_STA_ACPI	100	104	108	10C	180	184	188	18C
VGPIO_INT_CLR_ACPI	110	114	118	11C				

Note that the Address map is kept exactly the same as GPIO except for an additional two registers (last two) specific to the ACPI driver only.

9.14.1 VGPIO_LOCK_0

Lock bits are used to control the access to OE and OUT registers. When the lock bits are set, no one can write to OE and OUT bits. These can be programmed ONLY during Boot and get reset by chip reset only; that is, once set, the lock bits can be cleared only by applying chip reset.

VGPIO Port A - D Lock Bits (Read/Write)

Offset: 0x0..0xf | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxx00000000xxxxxxxx)

Bit	R/W	Reset	Description
15	RW	DISABLE	LOCK_7: Lock access to pin 7 OE and OUT 0 = DISABLE (i.e., OE and OUT bits unlocked, can be written) 1 = ENABLE (i.e., OE and OUT bits locked, cannot be written)
14	RW	DISABLE	LOCK_6: Lock access to pin 6 OE and OUT 0 = DISABLE 1 = ENABLE
13	RW	DISABLE	LOCK_5: Lock access to pin 5 OE and OUT 0 = DISABLE 1 = ENABLE
12	RW	DISABLE	LOCK_4: Lock access to pin 4 OE and OUT 0 = DISABLE 1 = ENABLE
11	RW	DISABLE	LOCK_3: Lock access to pin 3 OE and OUT 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
10	RW	DISABLE	LOCK_2: Lock access to pin 2 OE and OUT 0 = DISABLE 1 = ENABLE
9	RW	DISABLE	LOCK_1: Lock access to pin 1 OE OUT 0 = DISABLE 1 = ENABLE
8	RW	DISABLE	LOCK_0: Lock access to pin 0 OE and OUT 0 = DISABLE 1 = ENABLE
7	RO	X	RES_7: Lock access to each pin
6	RO	X	RES_6: Lock access to each pin
5	RO	X	RES_5: Lock access to each pin
4	RO	X	RES_4: Lock access to each pin
3	RO	X	RES_3: Lock access to each pin
2	RO	X	RES_2: Lock access to each pin
1	RO	X	RES_1: Lock access to each pin
0	RO	X	RES_0: Lock access to each pin

9.14.2 VGPIO_OE_0

The set of registers below is used to enable the VGPIO output interrupt (ACPI interrupt). The interrupt can be due to the VGPIO_OUT register or a virtual pin depending on the VGPIO_VIN_EN setting. This needs to be programmed depending upon whether the pin is either Input or Output.

This is an array of 4 identical register entries; the register fields below apply to each entry.

VGPIO Port A - D Output Enable Registers (Read/Write)

Offset: 0x10..0x1f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
6	0x0	BIT_6: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
5	0x0	BIT_5: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
4	0x0	BIT_4: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
3	0x0	BIT_3: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
2	0x0	BIT_2: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
1	0x0	BIT_1: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
0	0x0	BIT_0: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled

9.14.3 VGPIO_OUT_0

VGPIO_OE.x=1 (VGPIO output enabled) must be true for this register to be valid. This register will take affect only in VGPIO mode. This register is used to drive the output interrupt (ACPI interrupt).

This is an array of 4 identical register entries; the register fields below apply to each entry.

VGPIO Port A-D Output Value Registers (Read/Write)

Offset: 0x20.0x2f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH
6	0x0	BIT_6: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH
5	0x0	BIT_5: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH
4	0x0	BIT_4: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH
3	0x0	BIT_3: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH
2	0x0	BIT_2: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH
1	0x0	BIT_1: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH
0	0x0	BIT_0: VGPIO_VIN_EN.x=0 (in VGPIO mode) AND VGPIO_OE.x=1 (VGPIO output enabled) must be true for this to be a valid state. 0 = LOW 1 = HIGH

9.14.4 VGPIO_IN_0

This register stores the input pin value, used by the ACPI driver to copy the value from the PMIC pin.

This is an array of 4 identical register entries; the register fields below apply to each entry.

VGPIO Port A-D Input Value Registers (Read/Write)

Offset: 0x30..0x3f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: Input value 0 = LOW 1 = HIGH
6	0x0	BIT_6: Input value 0 = LOW 1 = HIGH

Bit	Reset	Description
5	0x0	BIT_5: Input value 0 = LOW 1 = HIGH
4	0x0	BIT_4 Input value 0 = LOW 1 = HIGH
3	0x0	BIT_3: Input value 0 = LOW 1 = HIGH
2	0x0	BIT_2: Input value 0 = LOW 1 = HIGH
1	0x0	BIT_1: Input value 0 = LOW 1 = HIGH
0	0x0	BIT_0: Input value 0 = LOW 1 = HIGH

9.14.5 VGPIO_INT_STA_0

All VGPIO inputs can be programmed to generate an interrupt request (CPU or ACPI). This configuration is also independent of that of the other pins.

In addition, the individual trigger-level for interrupt on each input can be programmed as either active-on-high or active-on-low. For example, to program an active-on-high interrupt on bit 3 of VGPIO-PORT_C, write '1' into bit 3 of the VGPIO_INT.LVL.C register (this sets the interrupt to be active-on-high), and then write '1' into bit 3 of VGPIO_INT.ENB.C (this enables interrupt on the named bit).

The interrupt flag status can be read in the appropriate bit of the VGPIO_INT.STA.C register by the CPU. When the programmed interrupt occurs, status should be cleared by writing into the appropriate bit of the VGPIO_INT.CLR.C register. Note that the interrupt generated will be routed to the processor only if the corresponding bit for VGPIO interrupts in the secondary interrupt controller is enabled.

The interrupt controls also apply to the Interrupt generation logic for the ACPI. All controls are shared except the enable ‘VGPIO_INT.ENB’. Instead the VGPIO_OEx is used to enable the interrupt for the ACPI. Also the flag status is read in the VGPIO_INT_STA_ACPI bits by the ACPI software. When the programmed interrupt occurs, status should be cleared by writing into the appropriate bit of the VGPIO_INT_CLR_ACPI register

VGPIO Port A-D Interrupt Status Registers

VGPIO_INT.ENB.x=1 must be true for this condition to be valid.Every VGPIO pin generates an Interrupt when switching from Low-High to High-Low.The Interrupt status for each port is saved in Interrupt status registers.

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x40..0x4f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
6	0x0	BIT_6: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
5	0x0	BIT_5: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE

Bit	Reset	Description
4	0x0	BIT_4: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
3	0x0	BIT_3: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
2	0x0	BIT_2: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
1	0x0	BIT_1: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
0	0x0	BIT_0: VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE

9.14.6 VGPIO_INT_ENB_0

Every VGPIO pin has an associated enable bit, which when enabled, routes the Interrupt to the CPU interrupt controller.

This is an array of 4 identical register entries; the register fields below apply to each entry.

VGPIO Port A-D Interrupt Enable Registers

Offset: 0x50..0x5f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE
6	0x0	BIT_6: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE
5	0x0	BIT_5: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE
4	0x0	BIT_4: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE
3	0x0	BIT_3: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE
2	0x0	BIT_2: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE
1	0x0	BIT_1: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE
0	0x0	BIT_0: Enables the interrupt for VGPIO_IN registers 0 = DISABLE 1 = ENABLE

9.14.7 VGPIO_INT_LVL_0

VGPIO can detect an interrupt for any Edge /Trigger/Level signals. The following register bits [7:0] are set when a Level signal toggles.

Bits [15:8] toggles whenever an Edge changes from High to Low.

Bits [15:8] toggles whenever an Edge changes from High to Low or Low to High

This is an array of 4 identical register entries; the register fields below apply to each entry

VGPIO Port A-D Interrupt Activation Level Registers

Offset: 0x60..0x6f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23	0x0	DELTA_7: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
22	0x0	DELTA_6: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
21	0x0	DELTA_5: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
20	0x0	DELTA_4: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
19	0x0	DELTA_3: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
18	0x0	DELTA_2: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
17	0x0	DELTA_1: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
16	0x0	DELTA_0: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
15	0x0	EDGE_7: 1 means Configure as Edge-Triggered Interrupt
14	0x0	EDGE_6: 1 means Configure as Edge-Triggered Interrupt
13	0x0	EDGE_5: 1 means Configure as Edge-Triggered Interrupt
12	0x0	EDGE_4: 1 means Configure as Edge-Triggered Interrupt
11	0x0	EDGE_3: 1 means Configure as Edge-Triggered Interrupt
10	0x0	EDGE_2: 1 means Configure as Edge-Triggered Interrupt
9	0x0	EDGE_1: 1 means Configure as Edge-Triggered Interrupt
8	0x0	EDGE_0: 1 means Configure as Edge-Triggered Interrupt
7	0x0	BIT_7: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
6	0x0	BIT_6: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
5	0x0	BIT_5: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
4	0x0	BIT_4: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
3	0x0	BIT_3: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
2	0x0	BIT_2: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
1	0x0	BIT_1: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
0	0x0	BIT_0: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH

9.14.8 VGPIO_INT_CLR_0

This write-only register clears the CPU Interrupts that are set. This is valid only in VGPIO mode and when VGPIO_INT.ENB is set.

This is an array of 4 identical register entries; the register fields below apply to each entry.

VGPIO Port A-D Interrupt (CPU) Flag Set-to-Clear Registers

Offset: 0x70..0x7f | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
6	0x0	BIT_6: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
5	0x0	BIT_5: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
4	0x0	BIT_4: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
3	0x0	BIT_3: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
2	0x0	BIT_2: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
1	0x0	BIT_1: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
0	0x0	BIT_0: VGPIO mode and VGPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR

Each register is provided with an individual 16-bit version for enabling Masked Writes and to avoid a Read-Modify-Write operation by the firmware. The exception is for the interrupt clear register, whose functionality is combined in the interrupt status register. Individual pins only can be programmed by enabling the write masks in the upper byte of these 16-bit registers.

VGPIO A-D Masked Output Enable (Masked Writes)

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x90..0x9f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
8	WO	0x0	MSK0: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
6	RW	0x0	BIT_6: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
5	RW	0x0	BIT_5: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
4	RW	0x0	BIT_4: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
3	RW	0x0	BIT_3: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
2	RW	0x0	BIT_2: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
1	RW	0x0	BIT_1: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled
0	RW	0x0	BIT_0: 0 = ACPI interrupt disabled 1 = ACPI interrupt enabled

9.14.9 VGPIO_MSK_OUT_0

VGPIO A-D Masked Output Enable (Masked Writes)

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0xa0..0xaf | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
8	WO	0x0	MSK0: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

9.14.10 VGPIO_MSK_IN_0

VGPIO A-D Masked Input Value Register (Masked Writes)

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0xb0..0xbf | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

9.14.11 VGPIO_MSK_INT_STA_0

VGPIO A-D Masked Interrupt Status (Masked Clears)

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0xc0..0xcf | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = IN_ACTIVE 1 = ACTIVE

Bit	R/W	Reset	Description
6	RW	0x0	BIT_6: 0 = IN_ACTIVE 1 = ACTIVE
5	RW	0x0	BIT_5: 0 = IN_ACTIVE 1 = ACTIVE
4	RW	0x0	BIT_4: 0 = IN_ACTIVE 1 = ACTIVE
3	RW	0x0	BIT_3: 0 = IN_ACTIVE 1 = ACTIVE
2	RW	0x0	BIT_2: 0 = IN_ACTIVE 1 = ACTIVE
1	RW	0x0	BIT_1: 0 = IN_ACTIVE 1 = ACTIVE
0	RW	0x0	BIT_0: 0 = IN_ACTIVE 1 = ACTIVE

9.14.12 VGPIO_MSK_INT_ENB_0

VGPIO A-D Masked Interrupt Enable (Masked Writes)

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0xd0..0xdf | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = DISABLE 1 = ENABLE
6	RW	0x0	BIT_6: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
5	RW	0x0	BIT_5: 0 = DISABLE 1 = ENABLE
4	RW	0x0	BIT_4: 0 = DISABLE 1 = ENABLE
3	RW	0x0	BIT_3: 0 = DISABLE 1 = ENABLE
2	RW	0x0	BIT_2: 0 = DISABLE 1 = ENABLE
1	RW	0x0	BIT_1: 0 = DISABLE 1 = ENABLE
0	RW	0x0	BIT_0: 0 = DISABLE 1 = ENABLE

9.14.13 VGPIO_INT_STA_ACPI_0

VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. Every VGPIO pin in the output direction generates an ACPI Interrupt when switching from Low to High or High to Low. The interrupt status for each port is saved in an Interrupt status registers.

VGPIO Port A-D Interrupt Status Registers

Offset: 0x100..0x10f | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
6	0x0	BIT_6: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
5	0x0	BIT_5: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
4	0x0	BIT_4: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
3	0x0	BIT_3: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
2	0x0	BIT_2: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
1	0x0	BIT_1: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
0	0x0	BIT_0: VGPIO_VIN_EN.x=1 or VGPIO_OEx=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE

9.14.14 VGPIO_INT_CLR_ACPI_0

This write-only register clears the ACPI Interrupts that are set. This register is valid only in VGPIO_OEx=1 or non-VGPIO mode (i.e., VGPIO_VIN_ENx =1).

VGPIO Port A-D Interrupt (ACPI) Flag Set-to-Clear Registers

Offset: 0x110..0x11f | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	BIT_7: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
6	0x0	BIT_6: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
5	0x0	BIT_5: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
4	0x0	BIT_4: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
3	0x0	BIT_3: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
2	0x0	BIT_2: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
1	0x0	BIT_1: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
0	0x0	BIT_0: VGPIO_OEx=1 or VGPIO_VIN_ENx =1 must be true for this condition to be valid. 0 = SET 1 = CLEAR

9.14.15 VGPIO_MSK_INT_STA_ACPI_0

VGPIO A-D Masked Interrupt Status (Masked Clears)

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x180..0x18f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
10	WO	0x0	MSK2: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for writes 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = IN_ACTIVE 1 = ACTIVE
6	RW	0x0	BIT_6: 0 = IN_ACTIVE 1 = ACTIVE
5	RW	0x0	BIT_5: 0 = IN_ACTIVE 1 = ACTIVE
4	RW	0x0	BIT_4: 0 = IN_ACTIVE 1 = ACTIVE
3	RW	0x0	BIT_3: 0 = IN_ACTIVE 1 = ACTIVE
2	RW	0x0	BIT_2: 0 = IN_ACTIVE 1 = ACTIVE
1	RW	0x0	BIT_1: 0 = IN_ACTIVE 1 = ACTIVE
0	RW	0x0	BIT_0: 0 = IN_ACTIVE 1 = ACTIVE

9.15 Pinmux Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

9.15.1 PINMUX_AUX_SDMMC1_CLK_0

Offset: 0x3000 | Read/Write: R/W | Reset: 0x00002074 (0bxxxxxxxxxxxxxxxxx010xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.2 PINMUX_AUX_SDMMC1_CMD_0

Offset: 0x3004 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.3 PINMUX_AUX_SDMMC1_DAT3_0

Offset: 0x3008 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.4 PINMUX_AUX_SDMMC1_DAT2_0

Offset: 0x300c | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.5 PINMUX_AUX_SDMMC1_DAT1_0

Offset: 0x3010 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.6 PINMUX_AUX_SDMMC1_DAT0_0

Offset: 0x3014 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.7 PINMUX_AUX_SDMMC3_CLK_0

Offset: 0x301c | Read/Write: R/W | Reset: 0x00002074 (0bxxxxxxxxxxxxxxxxx010xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.8 PINMUX_AUX_SDMMC3_CMD_0

Offset: 0x3020 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.9 PINMUX_AUX_SDMMC3_DAT0_0

Offset: 0x3024 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.10 PINMUX_AUX_SDMMC3_DAT1_0

Offset: 0x3028 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.11 PINMUX_AUX_SDMMC3_DAT2_0

Offset: 0x302c | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.12 PINMUX_AUX_SDMMC3_DAT3_0

Offset: 0x3030 | Read/Write: R/W | Reset: 0x00002078 (0bxxxxxxxxxxxxxxxxxx010xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.13 PINMUX_AUX_PEX_L0_RST_N_0

Offset: 0x3038 | Read/Write: R/W | Reset: 0x00000460 (0bxxxxxxxxxxxxxxxxxxxx001x001100000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	PASSTHROUGH	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE0	PM: 0 = PE0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.14 PINMUX_AUX_PEX_L0_CLKREQ_N_0

Offset: 0x303c | Read/Write: R/W | Reset: 0x00000470 (0bxxxxxxxxxxxxxxxxxxxx001x001110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE0	PM: 0 = PE0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.15 PINMUX_AUX_PEX_WAKE_N_0

Offset: 0x3040 | Read/Write: R/W | Reset: 0x00000470 (0bxxxxxxxxxxxxxxxxxxxx001x001110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE	PM: 0 = PE 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.16 PINMUX_AUX_PEX_L1_RST_N_0

Offset: 0x3044 | Read/Write: R/W | Reset: 0x00000460 (0bxxxxxxxxxxxxxxxxxxxx001x001100000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	PASSTHROUGH	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE1	PM: 0 = PE1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.17 PINMUX_AUX_PEX_L1_CLKREQ_N_0

Offset: 0x3048 | Read/Write: R/W | Reset: 0x00000470 (0bxxxxxxxxxxxxxxxxxxxx001x001110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PE1	PM: 0 = PE1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.18 PINMUX_AUX_SATA_LED_ACTIVE_0

Offset: 0x304c | Read/Write: R/W | Reset: 0x00000060 (0bxxxxxxxxxxxxxxxxxxxx00xx001100000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	PASSTHROUGH	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SATA	PM: 0 = SATA 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.19 PINMUX_AUX_SPI1_MOSI_0

Offset: 0x3050 | Read/Write: R/W | Reset: 0x0000e074 (0bxxxxxxxxxxxxxxxx1110xx0x01110100)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.20 PINMUX_AUX_SPI1_MISO_0

Offset: 0x3054 | Read/Write: R/W | Reset: 0x0000e074 (0bxxxxxxxxxxxxxxxx1110xx0x01110100)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.21 PINMUX_AUX_SPI1_SCK_0

Offset: 0x3058 | Read/Write: R/W | Reset: 0x0000e074 (0bxxxxxxxxxxxxxxxx1110xx0x01110100)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.22 PINMUX_AUX_SPI1_CS0_0

Offset: 0x305c | Read/Write: R/W | Reset: 0x0000e078 (0bxxxxxxxxxxxxxxxx1110xx0x01111000)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.23 PINMUX_AUX_SPI1_CS1_0

Offset: 0x3060 | Read/Write: R/W | Reset: 0x0000e078 (0bxxxxxxxxxxxxxxxx1110xx0x01111000)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.24 PINMUX_AUX_SPI2_MOSI_0

Offset: 0x3064 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = DTV 2 = RSVD2 3 = RSVD3

9.15.25 PINMUX_AUX_SPI2_MISO_0

Offset: 0x3068 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = DTV 2 = RSVD2 3 = RSVD3

9.15.26 PINMUX_AUX_SPI2_SCK_0

Offset: 0x306c | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = DTV 2 = RSVD2 3 = RSVD3

9.15.27 PINMUX_AUX_SPI2_CS0_0

Offset: 0x3070 | Read/Write: R/W | Reset: 0x00006078 (0bxxxxxxxxxxxxxxxxxx110xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = DTV 2 = RSVD2 3 = RSVD3

9.15.28 PINMUX_AUX_SPI2_CS1_0

Offset: 0x3074 | Read/Write: R/W | Reset: 0x00006078 (0bxxxxxxxxxxxxxxxxxx110xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.29 PINMUX_AUX_SPI4_MOSI_0

Offset: 0x3078 | Read/Write: R/W | Reset: 0x0000e074 (0bxxxxxxxxxxxxxxxxx1110xx0x01110100)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI4	PM: 0 = SPI4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.30 PINMUX_AUX_SPI4_MISO_0

Offset: 0x307c | Read/Write: R/W | Reset: 0x0000e074 (0bxxxxxxxxxxxxxxxx1110xx0x01110100)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI4	PM: 0 = SPI4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.31 PINMUX_AUX_SPI4_SCK_0

Offset: 0x3080 | Read/Write: R/W | Reset: 0x0000e074 (0bxxxxxxxxxxxxxxxx1110xx0x01110100)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI4	PM: 0 = SPI4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.32 PINMUX_AUX_SPI4_CS0_0

Offset: 0x3084 | Read/Write: R/W | Reset: 0x0000e078 (0bxxxxxxxxxxxxxxxx1110xx0x01111000)

Bit	Reset	Description
15	DISABLE	E_PREEMP: 0 = DISABLE 1 = ENABLE
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI4	PM: 0 = SPI4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.33 PINMUX_AUX_QSPI_SCK_0

Offset: 0x3088 | Read/Write: R/W | Reset: 0x00003078 (0bxxxxxxxxxxxxxxxxx011xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	ENABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	QSPI	PM: 0 = QSPI 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.34 PINMUX_AUX_QSPI_CS_N_0

Offset: 0x308c | Read/Write: R/W | Reset: 0x00003078 (0bxxxxxxxxxxxxxxxxxx011xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	ENABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	QSPI	PM: 0 = QSPI 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.35 PINMUX_AUX_QSPI_IO0_0

Offset: 0x3090 | Read/Write: R/W | Reset: 0x00003078 (0bxxxxxxxxxxxxxxxxxx011xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	ENABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	QSPI	PM: 0 = QSPI 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.36 PINMUX_AUX_QSPI_IO1_0

Offset: 0x3094 | Read/Write: R/W | Reset: 0x00003078 (0bxxxxxxxxxxxxxxxxxx011xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	ENABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	QSPI	PM: 0 = QSPI 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.37 PINMUX_AUX_QSPI_IO2_0

Offset: 0x3098 | Read/Write: R/W | Reset: 0x00003078 (0bxxxxxxxxxxxxxxxxxx011xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	ENABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	QSPI	PM: 0 = QSPI 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.38 PINMUX_AUX_QSPI_IO3_0

Offset: 0x309c | Read/Write: R/W | Reset: 0x00003078 (0bxxxxxxxxxxxxxxxxxx011xx0x01111000)

Bit	Reset	Description
14:13	DRIVE_2X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X

Bit	Reset	Description
12	ENABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	QSPI	PM: 0 = QSPI 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.39 PINMUX_AUX_DMIC1_CLK_0

Offset: 0x30a4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	DMIC1	PM: 0 = DMIC1 1 = I2S3 2 = RSVD2 3 = RSVD3

9.15.40 PINMUX_AUX_DMIC1_DAT_0

Offset: 0x30a8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DMIC1	PM: 0 = DMIC1 1 = I2S3 2 = RSVD2 3 = RSVD3

9.15.41 PINMUX_AUX_DMIC2_CLK_0

Offset: 0x30ac | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DMIC2	PM: 0 = DMIC2 1 = I2S3 2 = RSVD2 3 = RSVD3

9.15.42 PINMUX_AUX_DMIC2_DAT_0

Offset: 0x30b0 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DMIC2	PM: 0 = DMIC2 1 = I2S3 2 = RSVD2 3 = RSVD3

9.15.43 PINMUX_AUX_DMIC3_CLK_0

Offset: 0x30b4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DMIC3	PM: 0 = DMIC3 1 = I2S5A 2 = RSVD2 3 = RSVD3

9.15.44 PINMUX_AUX_DMIC3_DAT_0

Offset: 0x30b8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DMIC3	PM: 0 = DMIC3 1 = I2S5A 2 = RSVD2 3 = RSVD3

9.15.45 PINMUX_AUX_GEN1_I2C_SCL_0

Offset: 0x30bc | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C1	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.46 PINMUX_AUX_GEN1_I2C_SDA_0

Offset: 0x30c0 | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C1	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.47 PINMUX_AUX_GEN2_I2C_SCL_0

Offset: 0x30c4 | Read/Write: R/W | Reset: 0x00000572 (0bxxxxxxxxxxxxxxxxxxxx001x101110010)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD2	PM: 0 = I2C2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.48 PINMUX_AUX_GEN2_I2C_SDA_0

Offset: 0x30c8 | Read/Write: R/W | Reset: 0x00000572 (0bxxxxxxxxxxxxxxxxxxxx001x101110010)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD2	PM: 0 = I2C2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.49 PINMUX_AUX_GEN3_I2C_SCL_0

Offset: 0x30cc | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C3	PM: 0 = I2C3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.50 PINMUX_AUX_GEN3_I2C_SDA_0

Offset: 0x30d0 | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C3	PM: 0 = I2C3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.51 PINMUX_AUX_CAM_I2C_SCL_0

Offset: 0x30d4 | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C3	PM: 0 = I2C3 1 = I2CVI 2 = RSVD2 3 = RSVD3

9.15.52 PINMUX_AUX_CAM_I2C_SDA_0

Offset: 0x30d8 | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C3	PM: 0 = I2C3 1 = I2CVI 2 = RSVD2 3 = RSVD3

9.15.53 PINMUX_AUX_PWR_I2C_SCL_0

Offset: 0x30dc | Read/Write: R/W | Reset: 0x00000170 (0bxxxxxxxxxxxxxxxxxxxx000x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	DISABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2CPMU	PM: 0 = I2CPMU 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.54 PINMUX_AUX_PWR_I2C_SDA_0

Offset: 0x30e0 | Read/Write: R/W | Reset: 0x00000170 (0bxxxxxxxxxxxxxxxxxxxx000x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	DISABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2CPMU	PM: 0 = I2CPMU 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.55 PINMUX_AUX_UART1_TX_0

Offset: 0x30e4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTA	PM: 0 = UARTA 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.56 PINMUX_AUX_UART1_RX_0

Offset: 0x30e8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	UARTA	PM: 0 = UARTA 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.57 PINMUX_AUX_UART1_RTS_0

Offset: 0x30ec | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTA	PM: 0 = UARTA 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.58 PINMUX_AUX_UART1_CTS_0

Offset: 0x30f0 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTA	PM: 0 = UARTA 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.59 PINMUX_AUX_UART2_TX_0

Offset: 0x30f4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTB	PM: 0 = UARTB 1 = I2S4A 2 = SPDIF 3 = UART

9.15.60 PINMUX_AUX_UART2_RX_0

Offset: 0x30f8 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTB	PM: 0 = UARTB 1 = I2S4A 2 = SPDIF 3 = UART

9.15.61 PINMUX_AUX_UART2_RTS_0

Offset: 0x30fc | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTB	PM: 0 = UARTB 1 = I2S4A 2 = RSVD2 3 = UART

9.15.62 PINMUX_AUX_UART2_CTS_0

Offset: 0x3100 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTB	PM: 0 = UARTB 1 = I2S4A 2 = RSVD2 3 = UART

9.15.63 PINMUX_AUX_UART3_TX_0

Offset: 0x3104 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = SPI4 2 = RSVD2 3 = RSVD3

9.15.64 PINMUX_AUX_UART3_RX_0

Offset: 0x3108 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = SPI4 2 = RSVD2 3 = RSVD3

9.15.65 PINMUX_AUX_UART3_RTS_0

Offset: 0x310c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = SPI4 2 = RSVD2 3 = RSVD3

9.15.66 PINMUX_AUX_UART3_CTS_0

Offset: 0x3110 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = SPI4 2 = RSVD2 3 = RSVD3

9.15.67 PINMUX_AUX_UART4_TX_0

Offset: 0x3114 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTD	PM: 0 = UARTD 1 = UART 2 = RSVD2 3 = RSVD3

9.15.68 PINMUX_AUX_UART4_RX_0

Offset: 0x3118 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTD	PM: 0 = UARTD 1 = UART 2 = RSVD2 3 = RSVD3

9.15.69 PINMUX_AUX_UART4_RTS_0

Offset: 0x311c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTD	PM: 0 = UARTD 1 = UART 2 = RSVD2 3 = RSVD3

9.15.70 PINMUX_AUX_UART4_CTS_0

Offset: 0x3120 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTD	PM: 0 = UARTD 1 = UART 2 = RSVD2 3 = RSVD3

9.15.71 PINMUX_AUX_DAP1_FS_0

Offset: 0x3124 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.72 PINMUX_AUX_DAP1_DIN_0

Offset: 0x3128 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.73 PINMUX_AUX_DAP1_DOUT_0

Offset: 0x312c | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.74 PINMUX_AUX_DAP1_SCLK_0

Offset: 0x3130 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	I2S1	PM: 0 = I2S1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.75 PINMUX_AUX_DAP2_FS_0

Offset: 0x3134 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.76 PINMUX_AUX_DAP2_DIN_0

Offset: 0x3138 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.77 PINMUX_AUX_DAP2_DOUT_0

Offset: 0x313c | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.78 PINMUX_AUX_DAP2_SCLK_0

Offset: 0x3140 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.79 PINMUX_AUX_DAP4_FS_0

Offset: 0x3144 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4B	PM: 0 = I2S4B 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.80 PINMUX_AUX_DAP4_DIN_0

Offset: 0x3148 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4B	PM: 0 = I2S4B 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.81 PINMUX_AUX_DAP4_DOUT_0

Offset: 0x314c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4B	PM: 0 = I2S4B 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.82 PINMUX_AUX_DAP4_SCLK_0

Offset: 0x3150 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	I2S4B	PM: 0 = I2S4B 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.83 PINMUX_AUX_CAM1_MCLK_0

Offset: 0x3154 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH3	PM: 0 = EXTPERIPH3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.84 PINMUX_AUX_CAM2_MCLK_0

Offset: 0x3158 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH3	PM: 0 = EXTPERIPH3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.85 PINMUX_AUX_JTAG_RTCK_0

Offset: 0x315c | Read/Write: R/W | Reset: 0x00000068 (0bxxxxxxxxxxxxxxxxxxxx00xx001101000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	PASSTHROUGH	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	JTAG	PM: 0 = JTAG 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.86 PINMUX_AUX_CLK_32K_IN_0

Offset: 0x3160 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxx00xxx00xx)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

9.15.87 PINMUX_AUX_CLK_32K_OUT_0

Offset: 0x3164 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SOC	PM: 0 = SOC 1 = BLINK 2 = RSVD2 3 = RSVD3

9.15.88 PINMUX_AUX_BATT_BCL_0

Offset: 0x3168 | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	BCL	PM: 0 = BCL 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.89 PINMUX_AUX_CLK_REQ_0

Offset: 0x316c | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxx0xxx001xx00xx)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

9.15.90 PINMUX_AUX_CPU_PWR_REQ_0

Offset: 0x3170 | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxx0xxx001xx00xx)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

9.15.91 PINMUX_AUX_PWR_INT_N_0

Offset: 0x3174 | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxx0xxx001xx00xx)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

9.15.92 PINMUX_AUX_SHUTDOWN_0

Offset: 0x3178 | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxx0xxx001xx00xx)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

9.15.93 PINMUX_AUX_CORE_PWR_REQ_0

Offset: 0x317c | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxx0xxx001xx00xx)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

9.15.94 PINMUX_AUX_AUD_MCLK_0

Offset: 0x3180 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	AUD	PM: 0 = AUD (AUD_CLK maps to EXTPERIPH1) 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.95 PINMUX_AUX_DVFS_PWM_0

Offset: 0x3184 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = CLDVFS 2 = SPI3 3 = RSVD3

9.15.96 PINMUX_AUX_DVFS_CLK_0

Offset: 0x3188 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = CLDVFS 2 = SPI3 3 = RSVD3

9.15.97 PINMUX_AUX_GPIO_X1_AUD_0

Offset: 0x318c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = SPI3 3 = RSVD3

9.15.98 PINMUX_AUX_GPIO_X3_AUD_0

Offset: 0x3190 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = SPI3 3 = RSVD3

9.15.99 PINMUX_AUX_GPIO_PCC7_0

Offset: 0x3194 | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.100 PINMUX_AUX_HDMI_CEC_0

Offset: 0x3198 | Read/Write: R/W | Reset: 0x00000570 (0bxxxxxxxxxxxxxxxxxxxx001x101110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	CEC	PM: 0 = CEC 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.101 PINMUX_AUX_HDMI_INT_DP_HPD_0

Offset: 0x319c | Read/Write: R/W | Reset: 0x00000574 (0bxxxxxxxxxxxxxxxxxxxx001x101110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DP	PM: 0 = DP 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.102 PINMUX_AUX_SPDIF_OUT_0

Offset: 0x31a0 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPDIF	PM: 0 = SPDIF 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.103 PINMUX_AUX_SPDIF_IN_0

Offset: 0x31a4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPDIF	PM: 0 = SPDIF 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.104 PINMUX_AUX_USB_VBUS_EN0_0

Offset: 0x31a8 | Read/Write: R/W | Reset: 0x00000560 (0bxxxxxxxxxxxxxxxxxxxx001x101100000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	PASSTHROUGH	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.105 PINMUX_AUX_USB_VBUS_EN1_0

Offset: 0x31ac | Read/Write: R/W | Reset: 0x00000560 (0bxxxxxxxxxxxxxxxxxxxx001x101100000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
10	ENABLE	E_IO_HV: 0 = DISABLE 1 = ENABLE
8	ENABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	PASSTHROUGH	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.106 PINMUX_AUX_DP_HPD0_0

Offset: 0x31b0 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DP	PM: 0 = DP 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.107 PINMUX_AUX_WIFI_EN_0

Offset: 0x31b4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.108 PINMUX_AUX_WIFI_RST_0

Offset: 0x31b8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.109 PINMUX_AUX_WIFI_WAKE_AP_0

Offset: 0x31bc | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.110 PINMUX_AUX_AP_WAKE_BT_0

Offset: 0x31c0 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = UARTB 2 = SPDIF 3 = RSVD3

9.15.111 PINMUX_AUX_BT_RST_0

Offset: 0x31c4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = UARTB 2 = SPDIF 3 = RSVD3

9.15.112 PINMUX_AUX_BT_WAKE_AP_0

Offset: 0x31c8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.113 PINMUX_AUX_AP_WAKE_NFC_0

Offset: 0x31cc | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.114 PINMUX_AUX_NFC_EN_0

Offset: 0x31d0 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.115 PINMUX_AUX_NFC_INT_0

Offset: 0x31d4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.116 PINMUX_AUX_GPS_EN_0

Offset: 0x31d8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.117 PINMUX_AUX_GPS_RST_0

Offset: 0x31dc | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.118 PINMUX_AUX_CAM_RST_0

Offset: 0x31e0 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP1	PM: 0 = VGP1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.119 PINMUX_AUX_CAM_AF_EN_0

Offset: 0x31e4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VIMCLK	PM: 0 = VIMCLK 1 = VGP2 2 = RSVD2 3 = RSVD3

9.15.120 PINMUX_AUX_CAM_FLASH_EN_0

Offset: 0x31e8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VIMCLK	PM: 0 = VIMCLK 1 = VGP3 2 = RSVD2 3 = RSVD3

9.15.121 PINMUX_AUX_CAM1_PWDN_0

Offset: 0x31ec | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP4	PM: 0 = VGP4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.122 PINMUX_AUX_CAM2_PWDN_0

Offset: 0x31f0 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP5	PM: 0 = VGP5 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.123 PINMUX_AUX_CAM1_STROBE_0

Offset: 0x31f4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP6	PM: 0 = VGP6 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.124 PINMUX_AUX_LCD_TE_0

Offset: 0x31f8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DISPLAYA	PM: 0 = DISPLAYA 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.125 PINMUX_AUX_LCD_BL_PWM_0

Offset: 0x31fc | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DISPLAYA	PM: 0 = DISPLAYA 1 = PWM0 2 = SOR0 3 = RSVD3

9.15.126 PINMUX_AUX_LCD_BL_EN_0

Offset: 0x3200 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.127 PINMUX_AUX_LCD_RST_0

Offset: 0x3204 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.128 PINMUX_AUX_LCD_GPIO1_0

Offset: 0x3208 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DISPLAYB	PM: 0 = DISPLAYB 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.129 PINMUX_AUX_LCD_GPIO2_0

Offset: 0x320c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DISPLAYB	PM: 0 = DISPLAYB 1 = PWM1 2 = RSVD2 3 = SOR1

9.15.130 PINMUX_AUX_AP_READY_0

Offset: 0x3210 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.131 PINMUX_AUX_TOUCH_RST_0

Offset: 0x3214 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.132 PINMUX_AUX_TOUCH_CLK_0

Offset: 0x3218 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	TOUCH	PM: 0 = TOUCH (Note: TOUCH maps to TOUCH_CLK and EXTPERIPH2) 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.133 PINMUX_AUX_MODEM_WAKE_AP_0

Offset: 0x321c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.134 PINMUX_AUX_TOUCH_INT_0

Offset: 0x3220 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.135 PINMUX_AUX_MOTION_INT_0

Offset: 0x3224 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.136 PINMUX_AUX_ALS_PROX_INT_0

Offset: 0x3228 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.137 PINMUX_AUX_TEMP_ALERT_0

Offset: 0x322c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.138 PINMUX_AUX_BUTTON_POWER_ON_0

Offset: 0x3230 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.139 PINMUX_AUX_BUTTON_VOL_UP_0

Offset: 0x3234 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.140 PINMUX_AUX_BUTTON_VOL_DOWN_0

Offset: 0x3238 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.141 PINMUX_AUX_BUTTON_SLIDE_SW_0

Offset: 0x323c | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.142 PINMUX_AUX_BUTTON_HOME_0

Offset: 0x3240 | Read/Write: R/W | Reset: 0x00000078 (0bxxxxxxxxxxxxxxxxxxxx00xx001111000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.143 PINMUX_AUX_GPIO_PA6_0

Offset: 0x3244 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxx00xx000110000)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	DISABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	NONE	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SATA	PM: 0 = SATA 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.144 PINMUX_AUX_GPIO_PE6_0

Offset: 0x3248 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = I2S5A 2 = PWM2 3 = RSVD3

9.15.145 PINMUX_AUX_GPIO_PE7_0

Offset: 0x324c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = I2S5A 2 = PWM3 3 = RSVD3

9.15.146 PINMUX_AUX_GPIO_PH6_0

Offset: 0x3250 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.147 PINMUX_AUX_GPIO_PK0_0

Offset: 0x3254 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IQC0	PM: 0 = IQC0 1 = I2S5B 2 = RSVD2 3 = RSVD3

9.15.148 PINMUX_AUX_GPIO_PK1_0

Offset: 0x3258 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IQC0	PM: 0 = IQC0 1 = I2S5B 2 = RSVD2 3 = RSVD3

9.15.149 PINMUX_AUX_GPIO_PK2_0

Offset: 0x325c | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IQC0	PM: 0 = IQC0 1 = I2S5B 2 = RSVD2 3 = RSVD3

9.15.150 PINMUX_AUX_GPIO_PK3_0

Offset: 0x3260 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IQC0	PM: 0 = IQC0 1 = I2S5B 2 = RSVD2 3 = RSVD3

9.15.151 PINMUX_AUX_GPIO_PK4_0

Offset: 0x3264 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IQC1	PM: 0 = IQC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.152 PINMUX_AUX_GPIO_PK5_0

Offset: 0x3268 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IQC1	PM: 0 = IQC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.153 PINMUX_AUX_GPIO_PK6_0

Offset: 0x326c | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	IQC1	PM: 0 = IQC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.154 PINMUX_AUX_GPIO_PK7_0

Offset: 0x3270 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IQC1	PM: 0 = IQC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.155 PINMUX_AUX_GPIO_PL0_0

Offset: 0x3274 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.156 PINMUX_AUX_GPIO_PL1_0

Offset: 0x3278 | Read/Write: R/W | Reset: 0x00006074 (0bxxxxxxxxxxxxxxxxxx110xx0x01110100)

Bit	Reset	Description
14:13	DRIVE_4X	DRV_TYPE: 0 = DRIVE_1X 1 = DRIVE_2X 2 = DRIVE_3X 3 = DRIVE_4X
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
9	DISABLE	E_HSM: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SOC	PM: 0 = SOC 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.157 PINMUX_AUX_GPIO_PZ0_0

Offset: 0x327c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VIMCLK2	PM: 0 = VIMCLK2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.158 PINMUX_AUX_GPIO_PZ1_0

Offset: 0x3280 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE

Bit	Reset	Description
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VIMCLK2	PM: 0 = VIMCLK2 1 = SDMMC1 2 = RSVD2 3 = RSVD3

9.15.159 PINMUX_AUX_GPIO_PZ2_0

Offset: 0x3284 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = CCLA 2 = RSVD2 3 = RSVD3

9.15.160 PINMUX_AUX_GPIO_PZ3_0

Offset: 0x3288 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.161 PINMUX_AUX_GPIO_PZ4_0

Offset: 0x328c | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.15.162 PINMUX_AUX_GPIO_PZ5_0

Offset: 0x3290 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxx00xx001110100)

Bit	Reset	Description
12	DISABLE	E_SCHMT: 0 = DISABLE 1 = ENABLE
11	DISABLE	E_OD: DO NOT USE THIS FIELD. LEAVE IT AT THE DEFAULT VALUE. 0 = DISABLE 1 = ENABLE
8	DISABLE	E_LPDR: 0 = DISABLE 1 = ENABLE
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
5	PARKED	PARK: 0 = NORMAL 1 = PARKED
4	TRISTATE	TRISTATE: 0 = PASSTHROUGH 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NONE 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SOC	PM: 0 = SOC 1 = RSVD1 2 = RSVD2 3 = RSVD3

CHAPTER 10: ACTIVITY MONITOR

This chapter describes the activity monitor (ACTMON) block. The ACTMON acts as a repository for activity indicators from various units, performs averaging of past samples, and indicates any abnormality in activity levels by sending an interrupt to software. Devices provide an idle/busy signal to the ACTMON block.

10.1 Functional Description

Activity monitoring is a means to dynamically measure the utilization of units in the system to help drive software power management policies. In Tegra processors, Dynamic Voltage and Frequency Scaling (DVFS) is used as a power management mechanism. It uses utilization information from various units to select an efficient voltage and frequency that the unit should operate at, while providing the requisite performance. Voltage change decisions are handled by this central software agent, but the unit frequency can be modulated in one of the following ways:

- The unit controls its own frequency - no need for activity monitoring support.
- The device driver controls the unit frequency based on calculated workload or hints from a user-level driver – no need for activity monitoring support.
- The device driver controls the unit frequency based on utilization information from a “unit ACTMON”.
- The central DVFS agent controls the unit frequency based on utilization information from a “central ACTMON”.

In addition to tracking utilization, activity monitoring is also used to provide hardware support to make the process of activity monitoring interrupt-driven instead of polling-based. This support is in the form of averaging, watermark detection, and histogram hardware. This reduces software overhead in procuring and tracking utilization data.

The ACTMON block has the following features:

- One 32-bit counter for each signaling device that accumulates activity counts.
- Running average computation for each counter over past N samples (N=128).
- Watermark breach detection logic for each counter. This is used to detect abrupt change in utilization.
- Interrupt-driven operation.

The DVFS thread can directly access and use the activity monitors to compute the target frequency (and consequently, voltage) for each unit for the next sampling interval.

The following table summarizes the Tegra X1 activity monitors.

Table 34: Activity Monitors

Module	Counter Description (Counter Width = 32 Bits)
A7AVP	Number of system clock cycles that A7AVP is in the halt state
AHB	Number of AHB clock cycles when no data transfer is detected on the bus. Can select one specific master or all of them.
APB	Number of APB clock cycles when no data transfer is detected on the bus. Can select one specific master or all of them.
CPU	Statistics for CPU frequency. The flow controller asserts a signal when ALL the cores are halted.
MC-ALL	Number of Memory Controller clock cycles when any (including CCPLEX) memory access event is detected. The MC toggles a signal to ACTMON every ‘W’ cycles.
MC-CPU	Number of MC clock cycles when any memory access event from CCPLEX is detected. MC toggles a signal to ACTMON every ‘W’ cycles.
VIC	Number of video pipe cycles when the pipe is empty. This is a unit ACTMON.
NVENC	Number of pipe cycles when the pipe is empty or the engine is seeing memory bottlenecks. This is a unit ACTMON.

Table 34: Activity Monitors

Module	Counter Description (Counter Width = 32 Bits)
NVJPEG	Number of pipe cycles when the pipe is empty or the engine is seeing memory bottlenecks. This is a unit ACTMON.
NVDEC	Number of pipe cycles when the pipe is empty or the engine is seeing memory bottlenecks. This is a unit ACTMON.

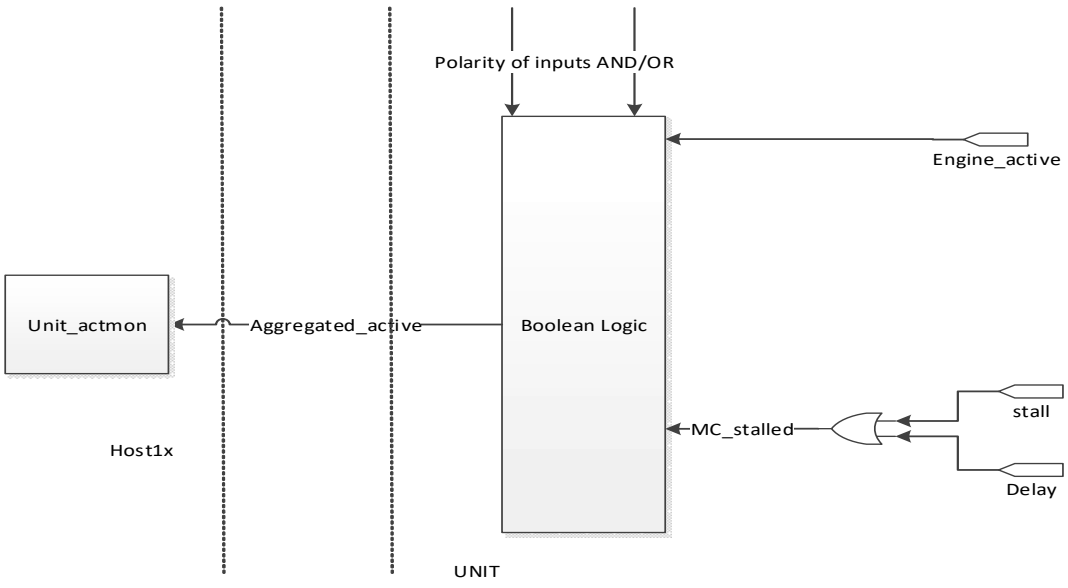
The table above assumes that the following units either run at a fixed frequency or do their own frequency management based on fixed workload, so they do not need central ACTMON support.

- VI/CSI (fixed)
- Display (workload-based, no monitors needed)
- Host (fixed)
- ISP (workload-based)
- NVENC (input-buffer based)
- VIC (utilization-based, uses per-unit ACTMON)
- SEC (fixed frequency)
- USB3 –Falcon (workload-based)
- MSELECT (frequency is based on other units and cannot be controlled independently)
- CPU Frequency: This ACTMON counts the CPU clock cycles. It has an edge detector and will count each edge of a divided CPU clock (1/1024). When the CPU is clock gated, the input will be zero. To prevent the ACTMON from breaching lower watermarks and interrupting software, software is expected to disable the ACTMON before going into clock gating or any other lower power state.

10.1.1 Enhancements to the Video Engine Unit ACTMONs for NVDEC/NVJPEG/NVENC

In Tegra X1 devices, unit ACTMONs for NVDEC/NVENC/NVJPG export an aggregated signal to the unit ACTMON in the Host1x controller as shown in the figure below.

Figure 13: Tegra X1 Video Engine Unit-ACTMON Implementation



The various signals in the above figure are described as follows:

- “Engine_active” goes HIGH for the ACTIVE cycle. This signal is asserted HIGH if any of the subunits of the engine is active.
- “Stall” goes HIGH when the engine is unable to issue a request to memory. (i.e., STALLED cycle)
- “Delay” goes HIGH when the engine is waiting for a read-response or write-acknowledge from memory.
- Hence, “MC_stalled” goes HIGH when the engine is stalled on memory or waiting for a response/acknowledge from memory.

The unit_ACTMON in the Host1x counts this Aggregated_active signal. The Aggregated active signal can be configured to be several things; a few examples are given below:

- Engine_active AND !(MC_stalled) – Only count cycles when the engine was active and not stalled on memory
- Engine_active – count cycles when the engine was active
- !(MC_stalled) – count cycles when the engine was not stalled on memory
- Engine_active AND MC_stalled – count cycles when the engine was active and stalled on memory

Devices such as MC, NVDEC, NVENC, NVJPG (unit ACTMON) operate at much higher frequencies than the ACTMON clock. Since ACTMON relies on idle/active pulses from these devices, a number of pulses will be dropped due to the slower ACTMON clock. To solve this problem, instead of sending active/idle for each clock, the device will toggle a signal for every ‘N’ active pulses of the device. (N is programmable). ACTMONs for such high-frequency devices will have an edge detector which will detect the toggle and update the corresponding counter by ‘N’. ‘N’ is the ratio of the device to ACTMON frequency and needs to be programmed by software into the client. Since the toggle rate of this signal is guaranteed to be slower than the ACTMON clock, no pulses will be missed.

The divided down CPU clock frequency is tracked using this ACTMON.

10.2 ACTMON Registers

Refer to “Reading Register Tables” in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

10.2.1 ACTMON GLB Status Register

10.2.1.1 ACTMON_GLB_STATUS_0

Offset: 0x0 | Read/Write: RO | Reset: 0xXX00XX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CPU_INTR: CPU Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
30	X	COP_INTR: COP Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
29	X	AHB_INTR: AHB Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
28	X	APB_INTR: APB Monitor Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
27	X	CPU_FREQ_INTR: CPU Frequency Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR
26	X	MCALL_INTR: MC_ALL Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR 1 = INTR

Bit	Reset	Description
25	X	MCCPU_INTR: MC_CPU Interrupt status. 1 = Interrupt detected; 0 = Interrupt not detected 0 = NOINTR: 1 = INTR
15	X	CPU_MON_ACT: CPU monitor active status. 1 = active 0 = INACTIVE 1 = ACTIVE
14	X	COP_MON_ACT: COP monitor active status. 1 = active 0 = INACTIVE 1 = ACTIVE
13	X	AHB_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
12	X	APB_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
10	X	CPU_FREQ_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
9	X	MCALL_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE
8	X	MCCPU_MON_ACT: 0 = inactive. 1 = active 0 = INACTIVE 1 = ACTIVE

10.2.2 ACTMON Global Period Register

10.2.2.1 ACTMON_GLB_PERIOD_CTRL_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SOURCE: 0 = MSEC: Sampling period time base in milliseconds 1 = USEC: Sampling period time base in microseconds
7:0	0x0	SAMPLE_PERIOD: Sampling period in milliseconds/microseconds (implemented as n+1 counter). Programming a 0 implies a 1 millisecond/microsecond sampling period.

10.2.3 CPU Registers

10.2.3.1 ACTMON_CPU_CTRL_0

Monitor Control Register

Offset: 0x80 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled

Bit	Reset	Description
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE: periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.3.2 ACTMON_CPU_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.3.3 ACTMON_CPU_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.3.4 ACTMON_CPU_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.3.5 ACTMON_CPU_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x90 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.3.6 ACTMON_CPU_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x94 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.3.7 ACTMON_CPU_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.3.8 ACTMON_CPU_COUNT_0

Monitor Status0 Register

Offset: 0x9c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.3.9 ACTMON_CPU_AVG_COUNT_0

Monitor Status1 Register

Offset: 0xa0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.3.10 ACTMON_CPU_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt; writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt; writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

Bit	Reset	Description
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect. 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

10.2.4 COP Registers

10.2.4.1 ACTMON_COP_CTRL_0

Monitor Control Register

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE: periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.4.2 ACTMON_COP_UPPER_WMARK_0

Upper Watermark Register

Offset: 0xc4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.4.3 ACTMON_COP_LOWER_WMARK_0

Lower Watermark Register

Offset: 0xc8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.4.4 ACTMON_COP_INIT_AVG_0

Initial AVG value, specified by software to set up the filter

Offset: 0xcc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.4.5 ACTMON_COP_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0xd0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.4.6 ACTMON_COP_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0xd4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.4.7 ACTMON_COP_COUNT_WEIGHT_0

Count Weight Register

Offset: 0xd8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.4.8 ACTMON_COP_COUNT_0

Monitor Status0 Register

Offset: 0xdc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.4.9 ACTMON_COP_AVG_COUNT_0

Monitor Status1 Register

Offset: 0xe0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.4.10 ACTMON_COP_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

10.2.5 AHB Registers

10.2.5.1 ACTMON_AHB_CTRL_0

Monitor Control Register

Offset: 0x100 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx1100000000000)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE: periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

Bit	Reset	Description
9:4	0x0	MONITOR_COND: Selection criteria on parent module for the activity signal, used only for APB/AHB monitors. Indicates which AHB master to monitor. All 1s means any master. 0 = CPU 1 = COP 2 = NA1 3 = CSITE 4 = ARC 5 = AHBDMA 6 = USB 7 = APBDMA 8 = NA2 9 = NA3 10 = NA4 11 = NA5 12 = NA6 13 = NA7 14 = SE 15 = DDS 16 = BSEA 17 = NA8 18 = USB2 19 = NA9 63 = ALL
3:0	0x0	SAMPLE_COND: Selection criteria on parent module for type of pulse signal, used only for APB/AHB monitor. 0 = AHB_MASTER_ACTIVE 1 = AHB_MASTER_SLAVE_ACTIVE 2 = AHB_DATA_XFER 3 = AHB_IDLE 4 = MASTER_IDLE 5 = AHB_BUSY 6 = DISABLE

10.2.5.2 ACTMON_AHB_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x104 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.5.3 ACTMON_AHB_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x108 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.5.4 ACTMON_AHB_INIT_AVG_0

Initial AVG value, specified by software to set up the filter

Offset: 0x10c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.5.5 ACTMON_AHB_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x110 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.5.6 ACTMON_AHB_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x114 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.5.7 ACTMON_AHB_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.5.8 ACTMON_AHB_COUNT_0

Monitor Status0 Register

Offset: 0x11c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.5.9 ACTMON_AHB_AVG_COUNT_0

Monitor Status1 Register

Offset: 0x120 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.5.10 ACTMON_AHB_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

Bit	Reset	Description
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

10.2.6 APB Registers

10.2.6.1 ACTMON_APB_CTRL_0

Monitor Control Register

Offset: 0x140 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx1100000000000)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by SW to enable sampling. Cleared in one of the following ways: (a) When SW intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE: periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period

Bit	Reset	Description
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.
9:4	0x0	MONITOR_COND: 0 = DTV 1 = I2C1 2 = I2C2 3 = I2C3 4 = I2C4 5 = DVC 6 = I2C6 7 = Unused, reserved 8 = Reserved 9 = SPI1 10 = SPI2 11 = SPI3 12 = SPI4 13 = SPI5 14 = SPI6 15 = QSPI 16 = UARTA 17 = UARTB 18 = UARTC 19 = UARTD 63 = ALL
3:0	0x0	SAMPLE_COND: Selection criteria on parent module for type of pulse signal, used only for APB/AHB monitor. 0 = PSEL_ACTIVE 1 = PREADY_ACTIVE 2 = PENABLE_PSEL_ACTIVE 3 = APB_IDLE 4 = DISABLE

10.2.6.2 ACTMON_APB_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x144 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.6.3 ACTMON_APB_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x148 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.6.4 ACTMON_APB_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x14c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.6.5 ACTMON_APB_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x150 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.6.6 ACTMON_APB_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x154 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.6.7 ACTMON_APB_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x158 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.6.8 ACTMON_APB_COUNT_0

Monitor Status0 Register

Offset: 0x15c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.6.9 ACTMON_APB_AVG_COUNT_0

Monitor Status1 Register

Offset: 0x160 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.6.10 ACTMON_APB_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x164 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

Bit	Reset	Description
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

10.2.6.11 ACTMON_APB_CTRL_SAPB_0

Monitor Interrupt Register

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:4	0x0	MONITOR_COND_SAPB: Selection criteria on parent module for the activity signal, used only for APB/AHB monitors. 0 = APB2JTAG 1 = APE 2 = ATOMICS 3 = CEC 4 = CSITE 5 = DDS 6 = DP2 7 = DVFS 8 = EMC0 9 = EMC1 10 = EMCB 11 = FUSE 12 = HDA 13 = KFUSE 14 = LA 15 = MC0 16 = MC1 17 = MCB 18 = MIPICAL 19 = MISC 20 = PINMUX_AUX 21 = PMC 22 = PWM 23 = RTC 24 = SATA 25 = SATA_AUX 26 = SDMMC1 27 = SDMMC2 28 = SDMMC3 29 = SDMMC4 30 = SE 31 = SOC_THERM 32 = SECURE_REGS 33 = STM 34 = SYSCTR0 35 = SYSCTR1 36 = XUSB_DEV 37 = XUSB_HOST 38 = SUXB_PADCTL 63 = ALL
3:0	0x0	SAMPLE_COND_SAPB: Selection criteria on parent module for type of pulse signal, used only for APB/AHB monitor. 0 = PSEL_ACTIVE 1 = PREADY_ACTIVE 2 = PENABLE_PSEL_ACTIVE 3 = APB_IDLE 4 = DISABLE

10.2.7 CPU Frequency Registers

10.2.7.1 ACTMON_CPU_FREQ_CTRL_0

Monitor Control Register

Offset: 0x180 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE: periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.7.2 ACTMON_CPU_FREQ_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x184 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.7.3 ACTMON_CPU_FREQ_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x188 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.7.4 ACTMON_CPU_FREQ_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x18c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.7.5 ACTMON_CPU_FREQ_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x190 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.7.6 ACTMON_CPU_FREQ_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x194 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.7.7 ACTMON_CPU_FREQ_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x198 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Weight value for count measurements.

10.2.7.8 ACTMON_CPU_FREQ_COUNT_0

Monitor Status0 Register

Offset: 0x19c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.7.9 ACTMON_CPU_FREQ_AVG_COUNT_0

Monitor Status1 Register

Offset: 0x1a0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.7.10 ACTMON_CPU_FREQ_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. a write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. a write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

10.2.8 MC_ALL Registers

10.2.8.1 ACTMON_MCALL_CTRL_0

Monitor Control Register

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.

Bit	Reset	Description
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE: periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.8.2 ACTMON_MCALL_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x1c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.8.3 ACTMON_MCALL_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x1c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.8.4 ACTMON_MCALL_INIT_AVG_0

Initial AVG value, specified by SW to set up the filter

Offset: 0x1cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.8.5 ACTMON_MCALL_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x1d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.8.6 ACTMON_MCALL_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x1d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.8.7 ACTMON_MCALL_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x1d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.8.8 ACTMON_MCALL_COUNT_0

Monitor Status0 Register

Offset: 0x1dc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.8.9 ACTMON_MCALL_AVG_COUNT_0

Monitor Status1 Register

Offset: 0x1e0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.8.10 ACTMON_MCALL_INTR_STATUS_0

Monitor Interrupt Register

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

Bit	Reset	Description
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

10.2.9 MC_CPU Registers

10.2.9.1 ACTMON_MCCPU_CTRL_0

Monitor Control Register

Offset: 0x200 | Read/Write: R/W | Reset: 0x00001800 (0b00000000000000xxxxx110xxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enable Monitor. Set by software to enable sampling. Cleared in one of the following ways: (a) When software intends to stop the monitor, it can do so by clearing this field, (b) when the sampling period expires (and we are not in the periodic mode) 0 = DISABLE 1 = ENABLE
30	0x0	CONSECUTIVE_ABOVE_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_UPPER_NUM' upper watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
29	0x0	CONSECUTIVE_BELOW_WMARK_EN: Enable interrupt when consecutive 'CONSECUTIVE_LOWER_NUM' lower watermark breaches are detected. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
28:26	0x0	CONSECUTIVE_ABOVE_WMARK_NUM: Number (N+1) of consecutive upper watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
25:23	0x0	CONSECUTIVE_BELOW_WMARK_NUM: Number (N+1) of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
22	0x0	WHEN_OVERFLOW_EN: Enable interrupt for the Number of consecutive lower watermark breaches that need to occur to raise an interrupt. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
21	0x0	AVG_ABOVE_WMARK_EN: Enable interrupt when AVG value is above its upper watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
20	0x0	AVG_BELOW_WMARK_EN: Enable interrupt when AVG value is below its lower watermark value. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
19	0x0	AT_END_EN: Enable interrupt at the end of sample period. 0 = DISABLE: interrupt is disabled. 1 = ENABLE: interrupt is enabled.
18	0x0	ENB_PERIODIC: Enable periodic mode. Sample for one sample period or periodic sampling 0 = DISABLE: periodic mode is disabled, samples for only 1 period 1 = ENABLE: periodic mode is enabled, keeps on sampling and updating value after every sample period
12:10	0x6	K_VAL: variable for IIR filter. Default is 6, which translates to 2^(K+1) = 128.

10.2.9.2 ACTMON_MCCPU_UPPER_WMARK_0

Upper Watermark Register

Offset: 0x204 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.9.3 ACTMON_MCCPU_LOWER_WMARK_0

Lower Watermark Register

Offset: 0x208 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.9.4 ACTMON_MCCPU_INIT_AVG_0

Initial AVG value, specified by software to set up the filter

Offset: 0x20c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

10.2.9.5 ACTMON_MCCPU_AVG_UPPER_WMARK_0

AVG Upper Watermark Register

Offset: 0x210 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Upper watermark for count value.

10.2.9.6 ACTMON_MCCPU_AVG_LOWER_WMARK_0

AVG Lower Watermark Register

Offset: 0x214 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Lower watermark for count value.

10.2.9.7 ACTMON_MCCPU_COUNT_WEIGHT_0

Count Weight Register

Offset: 0x218 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: weight value for count measurements.

10.2.9.8 ACTMON_MCCPU_COUNT_0

Monitor Status0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the number of active count pulses in one period

10.2.9.9 ACTMON_MCCPU_AVG_COUNT_0

Monitor Status1 Register

Offset: 0x220 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL: Indicates the AVG count value

10.2.9.10 ACTMON_MCCPU_INTR_STATUS_0

Monitor interrupt Register

Offset: 0x224 | Read/Write: R/W | Reset: 0x00000000 (0b000xx000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CONSECUTIVE_UPPER: Assert at the end of sampling period, if count value crosses upper watermark value consecutively for the number of times specified in CONSECUTIVE_UPPER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
30	0x0	CONSECUTIVE_LOWER: Assert at the end of sampling period, if count value crosses lower watermark value consecutively for the number of times specified in CONSECUTIVE_LOWER_NUM field. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
29	0x0	AT_END: Assert at the end of sampling period, if interrupt at end of sample period is enabled. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
26	0x0	WHEN_OVERFLOW: Assert at the end of sampling period, if there is an overflow. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
25	0x0	AVG_BELOW_WMARK: Assert at the end of sampling period, if AVG count value crosses lower AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected
24	0x0	AVG_ABOVE_WMARK: Assert at the end of sampling period, if AVG count value crosses upper AVG watermark value. A write value of 1 clears this interrupt, writing a 0 has no effect 0 = NOINTR: 0 = interrupt not detected 1 = INTR: 1 = interrupt detected

10.2.10 Histogram Registers

10.2.10.1 ACTMON_HISTOGRAM_CONFIG_0

Histogram Configuration

Offset: 0x300 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxx0000xxxxxxxxxxxx)

Bit	Reset	Description
15:12	NONE	SOURCE: 0 = NONE 1 = AHB 2 = APB 3 = COP 4 = CPU 5 = MC_ALL 6 = MC_CPU 7 = CPU_FREQ 8 = NA 9 = APB_MMIO
8:4	X	SHIFT: Scaling factor for the idle counter before the value is used to update histogram bucket.
3	X	STALL_ON_SINGLE_SATURATE: FALSE: continue incrementing buckets even when another bucket has saturated TRUE: stop incrementing bucket when at least one other bucket has saturated 0 = FALSE 1 = TRUE
2	X	NO_UNDERFLOW_BUCKET: FALSE: increase bucket 0 when idle time is less than the minimum value TRUE: ignore idle times that are less than the minimum value 0 = FALSE 1 = TRUE
1	X	LINEAR_MODE: DISABLE: bucket width increases exponentially with a power of two ENABLE: bucket width is the same for all buckets 0 = DISABLE 1 = ENABLE
0	X	ACTIVE: ENABLE: enable histogram recording 0 = DISABLE 1 = ENABLE

10.2.10.2 ACTMON_HISTOGRAM_CTRL_0

Offset: 0x304 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CLEAR: Clear all

10.2.10.3 ACTMON_HISTOGRAM_DATA_0

This is an array of 32 identical register entries; the register fields below apply to each entry.

Offset: 0x380..0x3ff | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE

CHAPTER 11: REAL-TIME CLOCK

The Real-Time Clock (RTC) module maintains seconds and milliseconds counters, and five alarm registers. The RTC is in the ‘always-on’ power domain, allowing for the counters to run and alarms to trigger when the system is in low-power state. If configured, interrupts triggered by the RTC can cause the system to wake up from a low-power state.

Features

- 10-bit milliseconds counter that runs off of a 32.768 kHz clock source.
- 32-bit seconds counter that increment for every 1000 milliseconds.
- Alarm feature that triggers an interrupt when the specified value matches the milliseconds counter.
- Alarm feature that triggers an interrupt when the specified value matches the seconds counter.
- Count-down alarm feature that triggers an alarm after counting down the specified number of seconds.
- Count-down alarm feature that triggers an alarm after counting down the specified number of milliseconds.
- Security bit that disables further processor writes to the seconds counter and ensures that the RTC clock keeps running.
- Hardware adjusts drift in clock which can occur due to PPM variations in oscillator output.

11.1 Functional Description

The RTC operates in two clock domains: the APB clock domain and the 32 kHz clock domain. The RTC continues updating the millisecond and second counters and continues triggering interrupts even when the MAIN partition is powered down. Since the APB clock is disabled when MAIN is powered down, registers are implemented in the 32 kHz clock domain.

All the registers except the BUSY register are implemented in the 32 kHz clock domain. Writes are transferred to the 32 kHz domain with BUSY.STATUS set to BUSY until the transfer is completed. Reads are shadowed in the APB clock domain and return immediately.

The RTC implements a Seconds Counter register, a Milliseconds Counter register, three Alarm registers, two countdown alarms, and various interrupt-related registers.

Writes to the seconds counter can be disabled by writing to the CONTROL.DIS_WR_SEC_CNT bit. Alarm registers and countdown alarm registers set interrupt bits when corresponding events occur. Interrupt status, mask, set and source registers reflect the status and also allow setting and clearing of various bits.

Resets

The RTC receives an asynchronous reset which is synchronized to the APB clock and RTC clock domains.

11.2 RTC Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

11.2.1 APBDEV_RTC_CONTROL_0

Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	WR_SEC_CNT: When set, writes to the SECONDS counter are disabled. Can only be cleared by resetting the RTC module 0 = DISABLE 1 = ENABLE

11.2.2 APBDEV_RTC_BUSY_0

Busy Register

Offset: 0x4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	STATUS: This bit is set when a write is initiated on the APB side. It is cleared once the write completes in RTC 32 kHz clock domain, which could be several thousands of APB clocks. This must be IDLE before a write is initiated. Note that this bit is only for writes. 0 = IDLE 1 = BUSY

11.2.3 APBDEV_RTC_SECONDS_0

Seconds Counter Register

The SECONDS register is copied over to the APB side every eight 32 kHz clocks (~250 µs). Because of this, performing a read immediately after a write might return an old value. This covers 49710.26 Days (or) 136.192 Years of 365 days each.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECONDS: The seconds counter is incremented every 1000 milliseconds.

11.2.4 APBDEV_RTC_SHADOW_SECONDS_0

Shadowed Seconds Counter Register

Shadow SECONDS register is updated over to the APB side whenever there is a read to milliseconds counter.

Since the software cannot read both registers at any given point of time, the Seconds register content is captured in this register. If software needs to read both seconds and milliseconds, read the MILLI_SECONDS register followed by a read of this register.

Offset: 0xc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SHADOW_SECONDS: A snapshot of the SECONDS counter is taken, whenever there is a read to MILLI_SECONDS Register.

11.2.5 APBDEV_RTC_MILLI_SECONDS_0

Milliseconds Counter Register

The Milliseconds register is copied over to the APB side every eight 32 kHz clocks (~250 µs). Because of this, performing a read immediately after a write might return an old value.

Offset: 0x10 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	MILLI_SECONDS: Milliseconds counter is incremented using the Bresenham algorithm

11.2.6 APBDEV_RTC_SECONDS_ALARM0_0

Seconds Alarm0 Registers

When the value in this register matches the seconds counter, the corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: Match value to trigger the alarm

11.2.7 APBDEV_RTC_SECONDS_ALARM1_0

Seconds Alarm1 Registers

When the value in this register matches the seconds counter, the corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: Match value to trigger the alarm

11.2.8 APBDEV_RTC_MILLI_SECONDS_ALARM_0

Milliseconds Alarm Register

When the value in this register matches the milliseconds counter, the corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	MSEC_MATCH_VALUE: Milliseconds match value.

11.2.9 APBDEV_RTC_SECONDS_COUNTDOWN_ALARM_0

Countdown Alarm Register

If ENABLE is set to ENABLED, an internal counter is loaded with VALUE and counted down once per second. The interrupt bit corresponding to the seconds countdown alarm (SEC_CDN_ALARM in the APBDEV_RTC_INTR_STATUS_0 register) is set after the specified number of seconds have elapsed. If REPEAT is ENABLED, the countdown operation is performed repeatedly.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	ENABLE: Enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: Repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: Number of seconds to countdown

11.2.10 APBDEV_RTC_MILLI_SECONDS_COUNTDOWN_ALARM_0

Countdown Alarm Register

If ENABLE is set to ENABLED, an internal counter is loaded with VALUE and counted down. The interrupt bit corresponding to the milliseconds countdown alarm (MSEC_CDN_ALARM in the APBDEV_RTC_INTR_STATUS_0 register) is set after the specified number of milliseconds have elapsed. If REPEAT is ENABLED, the countdown operation is performed repeatedly.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	ENABLE: Enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: Repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: Number of milliseconds to countdown

11.2.11 APBDEV_RTC_INTR_MASK_0

Interrupt Mask Register

This register stores the masks for the interrupts. If a bit is set 1 and the corresponding interrupt condition is satisfied, interrupt line to system interrupt controller is asserted.

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.12 APBDEV_RTC_INTR_STATUS_0

Interrupt Status Register

This register is RWC. Bits in it are high after the interrupt condition is satisfied. Any bits that are set to one in the data written to this register will clear their corresponding data bits. Any bits set to zero have no effect.

Offset: 0x2c | Read/Write: RWC | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.13 APBDEV_RTC_INTR_SOURCE_0

Interrupt Source Register

This read-only register returns the AND of the Interrupt Source and Interrupt Mask registers.

Offset: 0x30 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	MSEC_CDN_ALARM
3	X	SEC_CDN_ALARM
2	X	MSEC_ALARM
1	X	SEC_ALARM1
0	X	SEC_ALARM0

11.2.14 APBDEV_RTC_INTR_SET_0

Interrupt Set Register

Reserved. This write-only register is used for testing purposes only.

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.15 APBDEV_RTC_CORRECTION_FACTOR_0

Correction Factor (Digital Trimming) Register

Support is for +/- 500 ppm. The expected maximum deviation of the standard clock sources is +/- 50 ppm.

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

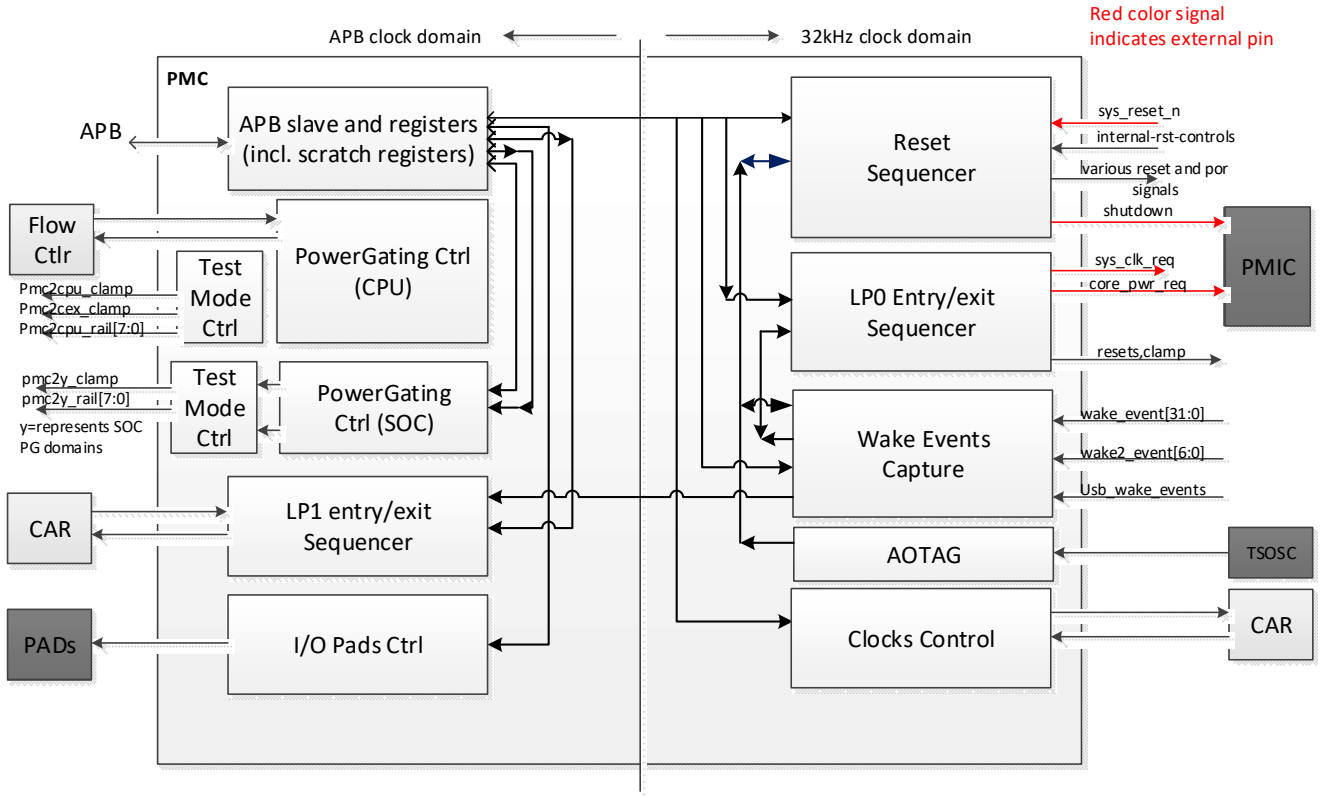
Bit	Reset	Description
9	0x0	DIRECTION: DECREMENT must be used when the 32 kHz clock is above its nominal frequency. 0 = DECREMENT 1 = INCREMENT
8:0	0x0	PPM

CHAPTER 12: POWER MANAGEMENT CONTROLLER

The Power Management Controller (PMC) block interacts with an external Power Manager Integrated Circuit (PMIC) through sideband signals. The PMC controls the entry and exit of the system from different sleep modes. It provides power-gating controllers for SoC and CPU power partitions except for the Maxwell GPU power partition. The PMC also provides deep power down (DPD) mode control for pads, and scratch storage to save some of the context during sleep modes (when the CPU and/or SoC power rails are off).

Sleep (LP1) and deep sleep (LP0) require specific logic to maintain some states and control the power domains, including signaling to the external PMIC to provide power to the main logic in the Tegra® X1 devices. All this logic is centralized in the PMC block.

Figure 14: PMC Block Diagram



Glossary

Term	Definition
AOTAG	Always-On Thermal Alarm Generator
Cold boot	The SoC partition power transitions from OFF to ON with no previous state available. Software must construct all states from scratch. Boot ROM is executed. DRAM is brought on-line.
HVC	Hardware Vmin Control: The hardware initiated flow to enter/exit the Vmin state on VDD_CPU. Now referred to as CC3.
LP0	Low Power 0 state in which DRAM is put in self-refresh, system state is saved in PMC + DRAM, the VDD_SOC and VDD_CPU rails are powered off, and the PMC is configured to monitor “LP0 exit wake events” which would trigger LP0 exit. This is also called the Deep Sleep state.
LP1	Low Power 1 state. Devices are power-gated, SoC clock domains are set to the minimum frequency (12 MHz and 38.4 MHz), the flow controller is configured to monitor “LP1 exit wake events,” DRAM is put in self-refresh, and the VDD_CPU rail is powered off. Also known as the Suspend state.
PMIC	Power Management IC (off-chip)
TSC	Timer’s System Counter (also known as Time Stamp Counter)
VDD_AO	Always ON power rail (also known as VDD_RTC power rail)
VDD_CPU	CPU power rail
VDD_SOC	SOC power rail (also known as VDD_CORE power rail)
Warm boot	Exit from LP0 state.

12.1 External Interface

Table 35: PMC External Interface Signals

Signal Name	Type	Signal Description
CLK_32K_OUT	Out	32 kHz clock out (active even during Deep Sleep). Can be used for systems that want a blinking LED during Deep Sleep
CORE_PWR_REQ	Out	Used to reach Vmin for CC3.
CPU_PWR_REQ	Out	
SHUTDOWN	Out	Shutdown indication from the Tegra processor to the platform.
CLK_REQ	Out	For systems where the system clock reference can be disabled during sleep
SYS_RESET_N	In	Hardware active-low reset signal. When this signal is asserted, the entire chip is reset including the PMC.
Wake-up Inputs	In	There are 47 pins designated as wake-up for triggering wake-up from Deep Sleep mode. These include 4 internal wake events from Deep Sleep mode for USB (UTMIP and UHSIC) and 1 internal wake event for XUSB.

12.1.1 Flow Controller Interface

The Tegra X1 flow controller manages CPU power and rail gating transitions. The flow controller sends CPU/non-CPU power-gating on/off or CPU rail on/off requests to the PMC. This interface is based on asynchronous req/ack. The flow controller explicitly requests (through cpu_id<>) which CPU it wants to be powered on/off.

See Chapter 18: Flow Controller for more information on CPU power gating.

12.2 Power Gating and Ungating

12.2.1 Clamp/Reset/Clock/PG-Enable Sequencing

12.2.2 Power Gating

For power-gating (powering off) a partition, the clamp, reset, clocks, and PG-enable (also known as sleep-enable) controls must be sequenced as shown in following diagram. For CCPLEX PG partitions, this sequencing is ensured by hardware (when

power-gating is done via the flow controller). For SoC (non-CCPLEX) PG partitions, this sequencing needs to be ensured by software.

The following registers are used for the Clamp/PG-Enable control of each PG partition:

- PMC_PWRGATE_TOGGLE
- PMC_REMOVE_CLAMPING_CMD

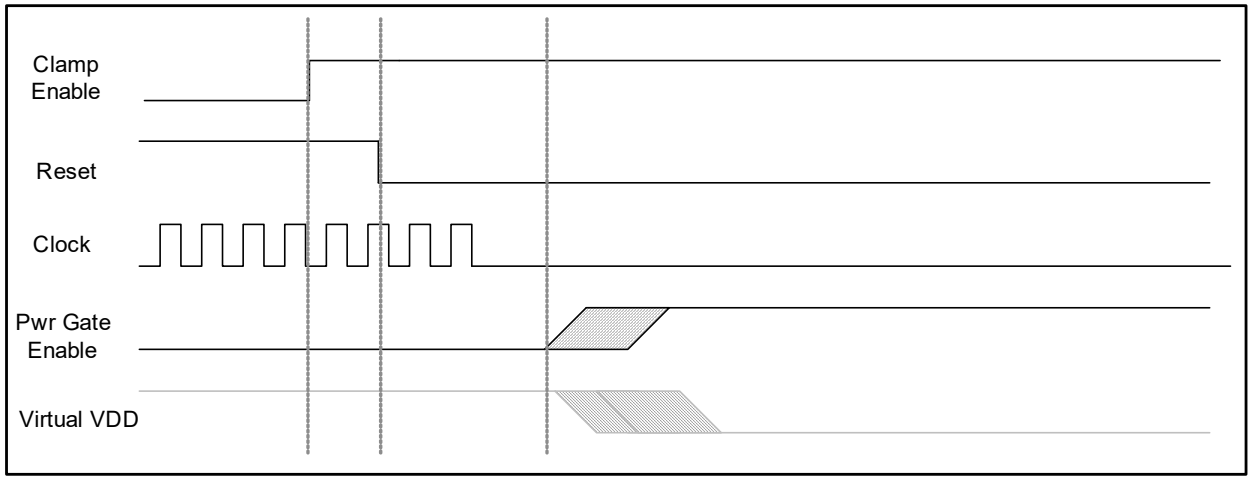
The following registers are used for the Clock/Reset control of each unit:

- RST_DEVICES_L/H/U/V/W/X
- CLK_OUT_ENB_L/H/U/V/W/X

In general, clamp-enable should be asserted before reset (as shown in the diagram). This works with synchronous or asynchronous clamping.

The figure below shows the required order of the various power-gating controls. Note that the reset shown here is an active low signal.

Figure 15: Power-Gating Timing Sequence for Clamp/Reset/Clock/PG-Enable

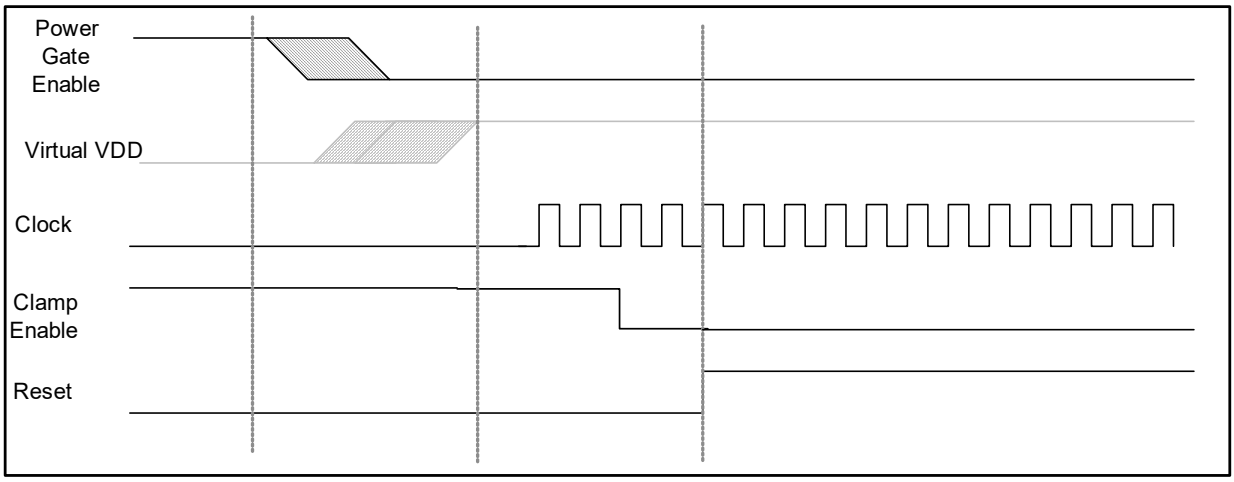


12.2.2.1 Power Ungating

For power-ungating (powering on) a partition, ensure that the clamp, reset, clocks, and PG-enable (sleep-enable) controls are sequenced as shown in the following diagram. For CCPLEX PG partitions, this sequencing is ensured by hardware (when power-gating is done via the flow controller). For SoC (non-CCPLEX) PG partitions, this sequencing needs to be ensured by software.

In general, the clamp enable should be deasserted before reset (as shown in the diagram). This works with synchronous or asynchronous clamping. To ensure that clocks have been brought up before any other activity progresses, software should add a 2us gap after the clock-activating command.

Figure 16: Power-Ungating Timing Sequence for Clamp/Reset/Clock/PG-Enable



12.2.3 Power Gating Zones

Both the PMC (power-controller for the SoC rail units) and the flow-controller (power-controller for the CPU rail units) divide each partition into 8 zones. Software configurable inter-zone timings are then provided. A single configurable timer is provided for all SOC rail partitions. When power-gating/ungating is triggered, the controllers sequence the pg-enable signals of consecutive zones with the programmed delays. The zones are powered up in one order and powered down in reverse order, but using the same inter-zone timings.

GPU power-gating is controlled by the GPMU unit internal to the GPU.

12.2.4 Context Save/Restore

For the CPU and SOC rail units, each unit's driver is responsible for saving and restoring any necessary state before and after a power-gating event. In the GPU, the GPMU is instead responsible for saving/restoring appropriate state.

12.2.5 Software Programming Guide

This section covers the software procedures for power gating and ungating the SOC rail power domains. Power gating and ungating of CPU power domains is not covered here as it is handled through Flow-Controller hardware unit.

Some clocks are routed to more than one power domain. If such a clock is disabled at the source due to the powering down of this source domain, the other domains depending on this clock will also be affected. Such clock dependencies are described here. Software has the responsibility to ensure that enabling or disabling of the clocks and domains is performed safely.

12.2.6 Power-Gating Policy

The power-management-framework coordinates the gating of all the SOC rail units. The high-level policy is defined below:

- A single kernel thread exists which has a 'power-management work-queue' data-structures for every unit.
- Any work request for a unit causes a reference count to be incremented in the work-queue for that unit. The unit's driver is responsible for requesting this.
- Any time work is completed, this work queue reference count must be decremented. The unit's driver may again request this decrement. This may be triggered by an interrupt from the unit hardware notifying that requested work has been completed.
- The reference counts are allowed to represent different kinds of work requests. For example, some could refer to register configuration accesses; others could be data (perhaps display frame) processing requests.
- A callback function into the power-management framework must be called any time the reference count reaches 0. This will cause a power-state transition to be considered.

- On reaching a reference count of 0, the unit must first be clock-gated. At the same time a timer must be set-off. When the timer reaches 0, and the unit is still idle, it must be power-gated.

The above flow necessitates that:

- All work requests for any power-gateable unit go through the reference-count tracking mechanism.
- The power-gating procedure appears atomic and is serialized with respect to any work requests.

If violated, a unit could be erroneously gated while still active OR the work request could be lost as the unit is already in the process of being gated.

The serialized and atomic requirement can be met by ensuring that when a work request is received while the power-gating procedure is in progress, power-gating the unit is either aborted or the work request delayed until the gating completes and the unit is then immediately un-gated.

12.2.6.1 ADSP

The ADSP unit in Tegra X1 may have some other module within the APE request work directly to it. However, since the ADSP is not power-gated independently to the rest of the APE (i.e. a single reference count exists for the whole APE), this does not create any problems. No other work requests, such as network audio wake events, are expected to be forwarded directly to the ADSP.

12.2.6.2 USB

Any wake events going to XUSB are routed through the non-power-gated PADCTL interrupt handling logic, subsequently handled by the XUSB PEP filter drivers. This ensures that work requests during power-gating don’t get lost.

12.2.7 Summary of Power-Gating Procedure

The general procedure for power gating an SOC power domain is as follows:

1. Write MC register to flush and block MC clients.

   Flush request is asserted during the hot-reset process, i.e. whenever a unit must be turned off. Software asserts the appropriate FLUSH_ENABLE bit in the MC_CLIENT_HOTRESET_CTRL_0 register. It then polls the MC_CLIENT_HOTRESET_STATUS_0 register for a “1” in the appropriate bit. When the register reads back 1, the pipe has been flushed and the unit can be power-gated or reset.
2. Write CAR register to assert unit resets.
3. Write CAR register to disable clocks.
4. Write PMC register to gate power domain (PMC will assert clamps itself before power-gating the unit).

The general procedure for power ungating of an SOC power domain is as follows:

1. Write PMC register to ungate power domain.
2. Write CAR register to enable clocks.
3. Write PMC register to disable clamps and poll CLAMP_STATUS to detect completion.
4. Write CAR register to de-assert resets.
5. Write MC register to enable MC clients.

In some cases, partitions require specific deviations from the general procedure due to various reasons. Deviations will be covered in the domain specific sections below.

Note: *Software must ensure that a unit is not already power-gated before attempting to power-gate it and that it is not already un-gated before attempting to un-gate it. This can be done by polling the unit specific APBDEV_PMC_PWRGATE_STATUS_0 register.*

Note: *Software must ensure that when a unit is gated no other units will be capturing the outputs of the unit to be gated. This is to ensure that the output meta-stability possible due to the asynchronous assertion of clamps and resets does not affect any neighboring units.*

12.2.7.1 Procedures for DIS Power Domain

DIS contains DisplayA module, and PLLD PLL. Power partitions DIS and DISM share the same PMC control. The two physical partitions are treated as a single logical partition called DIS. For any DSI output use-case, PLLD is needed. PLLD is the only PLL capable of generating differential clocks needed for DSI brick and is the only PLL connected to the brick as well. PLLD is on an un-power-gated supply. It stays running even if the corresponding partition is power-gated.

DIS Power Gating

1. Flush MC client DC by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.DC_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:
   a. DC_HOTRESET_STATUS
3. Set the following bits to assert reset to DIS:
   a. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP1_RST
4. Clear the following bits to disable clocks to DIS:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP1
5. Shut down PLLD (only if it’s not needed by any other unit)
6. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = DIS
   b. START = ENABLE
7. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit DIS is clear.

DIS Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = DIS
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit DIS is set
3. Reprogram PLLD and enable it (assuming PLLD is used for DisplayA and was not enabled already)
4. Set the following bits to enable clocks to DIS:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP1
5. Remove power gating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.DIS
6. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit DIS is cleared
7. Clear the following bits to de-assert reset to DIS:
   a. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP1_RST
8. Enable MC client DC by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.DC_FLUSH_ENABLE

12.2.7.2 Procedures for DISB Power Domain

DISB contains DisplayB module, and PLLD2 and PLLDP PLLs. The logical PMC partition “DISB” controls both DISB and DISBM physical partitions. The sequence is the same as DIS above, but also use the following:

- DISP2 for CAR reset and clock enable
- DCB for MC hot reset, and DISB for PMC
- PLLD2/PLLDP instead of PLLD

PLLD2/DP are on an un-power-gated supply. They stay running even if the corresponding partition is power-gated.

12.2.7.3 Procedures for SOR Power Domain

The SOR partition contains SOR, SOR1, DPAUX, DPAUX1, DSI, DSIB, CSICIL, and MIPI_CAL modules. For any DSI output use-case, PLLD in the DIS partition is needed. PLLD is the only PLL capable of generating differential clocks needed for DSI brick and is the only PLL connected to the brick as well.

> ***Note:*** *For any access to DPAUX registers, it is required that Host1x is clocked and out of reset, this is assumed to already be completed in the SOR sequence below.*

SOR Power Gating

1. If desired, perform SOR or SOR1 shutdown sequence. This is not strictly necessary as reset, below, will power down the pad.
2. Set the following bits to assert reset to all modules:
   a. CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_SOR0_RST
   b. CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_SOR1_RST
   c. CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_DPAUX_RST
   d. CLK_RST_CONTROLLER_RST_DEVICES_Y_0.SWR_DPAUX1_RST
   e. CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_MIPI_CAL_RST
   f. CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_CSI_RST
   g. CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_DSI_RST
   h. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_DSIB_RST
3. Clear the following bits to disable clocks to all modules:
   a. <same modules as above>
4. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = SOR
   b. START = ENABLE
5. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit SOR is clear

SOR Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = SOR
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit SOR is set
3. Program clock sources, and set the following bits to enable clocks to DIS:
   a. <same modules as above>

4. Remove powergating clamps by writing a 1 to the following bits:
    a. APBDEV_PMC_REMOVE_CLAMPING_CMD.SOR
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit SOR is cleared
6. Clear the following bits to de-assert reset to required SOR modules:
    a. <subset of modules above>
7. If DPAUX or DPAUX1 is not needed, power down the respective AUX pad.
    a. HYBRID_SPARE.PAD_PWR=POWERDOWN

Once the SOR partition is power-gated (which contains the internal HDA codec for HDMI/DP audio), the codec disappears from HDA link. The HDA controller sits in the APB partition so it will be kept alive. When SOR is powered back up, the HDA driver must assert and then de-assert the HDA link reset (CRST) and take the codec through the enumeration sequence. Otherwise, the codec would not be assigned an address (since the address would be lost during the power gating cycle). If this is not done, there is likely a chance that the codec would not reliably respond to commands after power un-gating.

Minimal Sequence when Programming the DPAUX Control for I2C6

If only I2C6 is required, follow this sequence to program the DPAUX Pad Control Register:

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
    a. PARTID = SOR
    b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the SOR bit is cleared
3. Set the following bits to enable clocks to DPAUX:
    a. CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_DPAUX
    b. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_HOST1X (only required if Host1x is not already out of reset and clocked)
4. Remove power-gating clamps by writing a 1 to the following bit:
    a. APBDEV_PMC_REMOVE_CLAMPING_CMD_0.SOR
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the SOR bit is cleared
6. Clear the following bits to deassert the reset to DPAUX:
    a. CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_DPAUX_RST
    b. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_HOST1X_RST (only required if Host1x is not already out of reset and clocked)

To enable I2C on DPAUX:

1. Enable Module Clock for DPAUX.
2. Release Module Reset for DPAUX.
3. Enable Clock for HOST1X.
4. Release HOST1X reset.
5. Enable Module Clock for I2C6.
6. Release Module Reset for I2C6.
7. Program MODE=I2C, I2C_SDA_INPUT_RCV=ENABLE, I2C_SCL_INPUT_RCV=ENABLE in DPAUX_HYBRID_PADCTL_0 register. (RTRegWr32(NV_ADDRESS_MAP_DPAUX_BASE + DPAUX_HYBRID_PADCTL_0 * 4, 0x0000C001);)
8. Program PAD_PWR =POWERUP in DPAUX_HYBRID_SPARE_0 register.

12.2.7.4 Procedures for MPEA Power Domain

From a software perspective, MPEA and MPEB are one logical partition. MPEA and MPEB power partitions share the same power gate timers.

The PMC also has an option to sequence both partitions at the same time to shorten power-gating latency. The TD_PWRGATE_INTER_PART_TIMER[3:0] register can be programmed to configure the period between two partitions power-gate up/down sequence. Partitions are powered up in order – MPEA, followed by MPEB. Power gating down is executed in reverse order. Programming a 0 value in the TD_PWRGATE_INTER_PART_TIMER register will trigger simultaneous power gating of both partitions.

Note: *There is a power domain (MPE) made up of two partitions: MPEA (NVENC Partition A) and MPEB (NVENC Partition B). See Section 12.4.6: SoC Rail Partitions for details.*

MPEA Power Gating

1. Flush MC client NVENC by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.NVENC_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:
   a. NVENC_HOTRESET_STATUS
3. Set the following bits to assert reset to NVENC:
   a. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_NVENC_RST
4. Clear the following bits to disable clocks to NVENC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_NVENC
5. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = MPEA
   b. START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit MPEA is clear.

MPEA Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = MPEA
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit MPEA is set:
3. Set the following bits to enable clocks to NVENC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_NVENC
4. Remove powergating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.MPEA
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit MPE is cleared.
6. Clear the following bits to de-assert reset to NVENC:
   b. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_NVENC_RST
7. Enable MC client NVENC by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.NVENC_FLUSH_ENABLE

12.2.7.5 Procedures for MPEB Power Domain

This partition is in the same power-domain as MPEA. The procedure previously described for MPEA therefore implicitly works for this partition too.

12.2.7.6 Procedures for XUSBB Power Domain

XUSBB Power Gating

1. Flush MC client XUSB_DEV by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.XUSB_DEV_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:
   a. XUSB_DEV_HOTRESET_STATUS
3. Set the following bits to assert reset to XUSB_DEV:
   a. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_DEV_RST
4. Clear the following bits to disable clocks to XUSB_DEV:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_DEV
5. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = XUSBB
   b. START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit XUSBB is clear.

XUSBB Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = XUSBB
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit XUSBB is set
3. Set the following bits to enable clocks to XUSB_DEV:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_DEV
4. Remove powergating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.XUSBB
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit XUSBB is cleared
6. Clear the following bits to de-assert reset to XUSB_DEV:
   a. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_DEV_RST
7. Enable MC client XUSB_DEV by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.XUSB_DEV_FLUSH_ENABLE

12.2.7.7 Procedures for XUSB Power Domain

See Section 22.8.7: XUSB Controller Power Gating.

12.2.7.8 Procedures for PCX Power Domain

See Section 34.3.5: ELPG.

12.2.7.9 Procedures for SAX Power Domain

SAX Power Gating

1. Program the following XUSB_PADCTL registers to put UPHY to IDDQ:
   a. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ_OVRD]:= 1
   b. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ]:= 1
   c. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ_OVRD]:= 1
   d. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ]:= 1
2. Program the following XUSB_PADCTL registers to put UPHY PAD PLL to IDDQ:
   a. XUSB_PADCTL_UPHY_PLL_S0_CTL_1_0[PLL0_PWR_OVRD]:= 1
   b. XUSB_PADCTL_UPHY_PLL_S0_CTL_1_0[PLL0_IDDQ]:= 1
3. Check the assertion status of DEVSLP and set the DEVSLP override with the following SATA AUX registers accordingly.
   a. $DEVSLP:= SATA_AUX_RX_STAT_INT_0[SATA_DEVLSP]
   b. SATA_AUX_MISC_CTRL_1_0[DEVSLP_OVERRIDE]:= '$DEVSLP'
4. Set the following CAR register bits to '1' to assert reset to SATA
   a. CLK_RST_CONTROLLER_RST_DEV_V_SET_0[SET_SATA_RST]:= 1
   b. CLK_RST_CONTROLLER_RST_DEV_V_SET_0[SET_SATA_OOB_RST]:= 1
   c. CLK_RST_CONTROLLER_RST_DEV_W_SET_0[SET_SATACOLD_RST]:= 1
5. Set the following CAR register bits to '1' to disable the clocks to individual SATA partitions.
   a. CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0[CLR_CLK_ENB_SATA]:= 1
   b. CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0[CLR_CLK_ENB_SATA_OOB]:= 1
6. System Power Management driver disables the SATA power rails.
   a. APBDEV_PMC_PWRGATE_TOGGLE_0[START]:= 1
   b. APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID]:= 'SAX'

Power Ungating

1. System Power Management driver enables the SATA power rails.
   a. APBDEV_PMC_PWRGATE_TOGGLE_0[START]:= 1
   b. APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID]:= 'SAX'
2. Read the following PMC register bits to confirm the power gating status of SATA
   a. APBDEV_PMC_PWRGATE_STATUS_0[SAX] == 'ON'
3. Set the following CAR register bits to '1' to enable the clocks to the SATA partition.
   a. CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA]:= 1
   b. CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA_OOB]:= 1
4. Set the following PMC register bits to '1' to remove the power clamps to the SATA partitions.
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [SAX]:= 1

      Read the following PMC register bits to confirm the power clamps to the SATA partition are removed.

   a. APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [SAX] == '0'
5. Set the following CAR register bits to '1' to de-assert reset to SATA
   a. CLK_RST_CONTROLLER_RST_DEV_V_CLR_0[CLR_SATA_RST]: = 1

b. CLK_RST_CONTROLLER_RST_DEV_V_CLR_0[CLR_SATA_OOB_RST]:= 1

c. CLK_RST_CONTROLLER_RST_DEV_W_CLR_0[CLR_SATACOLD_RST]:= 1

6. If DEVSLP is asserted, de-assert DEVSLP via the following SATA AUX register.
   a. SATA_AUX_MISC_CTRL_1_0[DEVSLP_OVERRIDE]:= '0'
7. Program the following XUSB_PADCTL registers to bringup up UPHY from IDDQ:
   a. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ_OVRD]:= 0
   b. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ]:= 0
   c. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ_OVRD]:= 0
   d. XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ]:= 0
8. Program the following XUSB_PADCTL registers to bringup UPHY PAD PLL from IDDQ:
   a. XUSB_PADCTL_UPHY_PLL_S0_CTL_1_0[PLL0_PWR_OVRD]:= 0
   b. XUSB_PADCTL_UPHY_PLL_S0_CTL_1_0[PLL0_IDDQ]:= 0

12.2.7.10 Procedures for NVDEC and NVJPG Power Domains

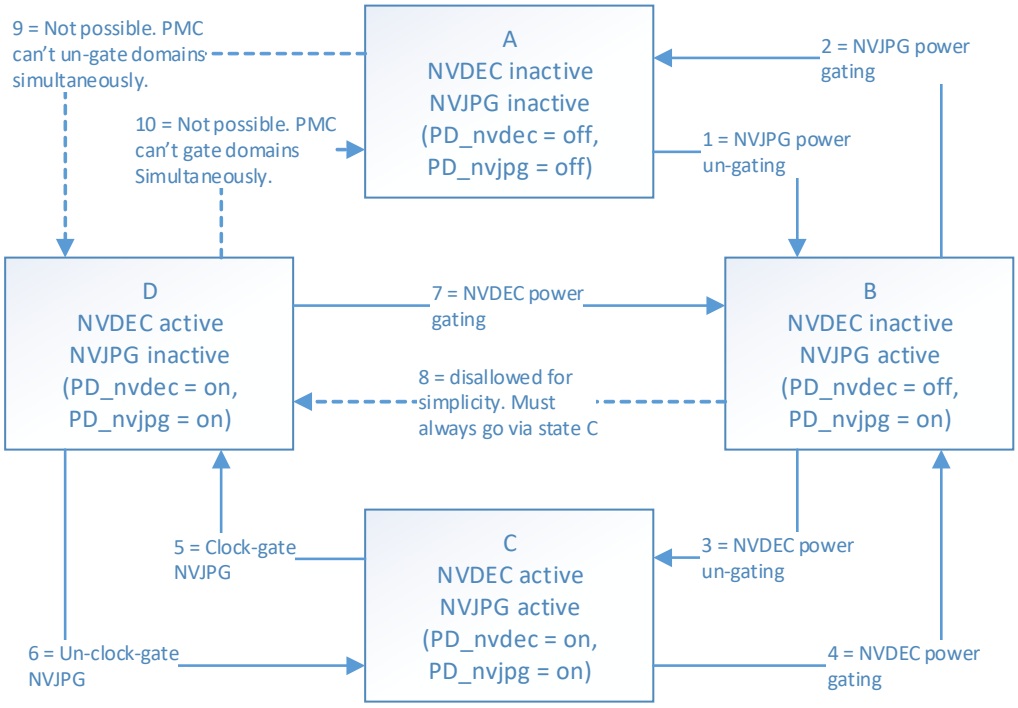
NVJPG contains the units NVJPG and NVDEC/Part A. This implies:

- When only NVDEC is required, such as in a VideoPlayback use case, NVJPG needs to be kept active.
- When only NVJPG is required, such as a gallery view, NVDEC part a needs to be kept active.This has the following additional requirements:

Anytime PD_NVDEC is required, PD_NVJPG must first be powered on. PD_NVJPG must not be power-gated off until after PD_NVDEC has been gated off. To minimize power, when the NVDEC unit is required but NVJPG is not required, NVJPG must be clock-gated off.

The permitted states and transitions are defined in the figure below. The transition details are described in the subsequent sections.

Figure 17: Allowed power states for NVDEC and NVJPG



NVJPG Power Gating

NVDEC has been power gated.

1. Flush MC client NVJPG by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_1_0.NVJPG_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_1_0 until the following bits are set:
   a. NVJPG_HOTRESET_STATUS
3. Set the following bits to assert reset to NVJPG:
   a. CLK_RST_CONTROLLER_RST_DEVICES_Y_0.SWR_NVJPG_RST
4. Clear the following bits to disable clocks to NVJPG:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVJPG
5. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = NVJPG
   b. START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit NVJPG is clear.

NVJPG Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = NVJPG
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit NVJPG is set
3. Set the following bits to enable clocks to NVJPG/NVDEC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVJPG
   b. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVDEC
4. Clear the following bits to disable clocks to NVDEC/NVJPG:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVDEC
   b. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVJPG (if video decode)
5. Remove powergating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.NVJPG
6. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit NVJPG is cleared
7. Clear the following bits to de-assert reset to NVJPG:
   a. CLK_RST_CONTROLLER_RST_DEVICES_Y_0.SWR_NVJPG_RST
8. Enable MC client NVJPG by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.NVJPG_FLUSH_ENABLE

NVDEC Power Gating

1. Flush MC client NVDEC by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_1_0.NVDEC_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_1_0 until the following bits are set:
   a. NVDEC_HOTRESET_STATUS
3. Set the following bits to assert reset to NVDEC:
   a. CLK_RST_CONTROLLER_RST_DEVICES_Y_0.SWR_NVDEC_RST

4. Clear the following bits to disable clocks to NVDEC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVDEC
5. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = NVDEC
   b. START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit NVDEC is clear.

NVDEC Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = NVDEC
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit NVDEC is set
3. Set the following bits to enable clocks to NVDEC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVDEC
4. Clear the following bits to disable clocks to NVDEC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_NVDEC
5. Remove powergating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.NVDEC
6. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit NVDEC is cleared
7. Clear the following bits to de-assert reset to NVDEC:
   a. CLK_RST_CONTROLLER_RST_DEVICES_Y_0.SWR_NVDEC_RST
8. Enable MC client NVDEC by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.NVDEC_FLUSH_ENABLE

12.2.7.11 Procedures for NVDECB and NVDECC Power Domains

These partitions contain the NVDEC unit in the PD_NVDEC power domain and so are described in the section above.

12.2.7.12 Procedures for VIC Power Domain

VIC Power Gating

1. Flush MC client VIC by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.VIC_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:
   a. VIC_HOTRESET_STATUS
3. Set the following bits to assert reset to VIC:
   a. CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_VIC_RST
4. Clear the following bits to disable clocks to VIC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_VIC
5. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = VIC
   b. START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit VIC is clear

VIC Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = VIC
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit VIC is set
3. Set the following bits to enable clocks to VIC:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0.CLK_ENB_VIC
4. Remove powergating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.VIC
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit VIC is cleared
6. Clear the following bits to de-assert reset to VIC:
   a. CLK_RST_CONTROLLER_RST_DEVICES_X_0.SWR_VIC_RST
7. Enable MC client VIC by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.VIC_FLUSH_ENABLE

12.2.7.13 Procedures for VICB VICC VICD Power Domains

All these partitions are in the same power-domain as VICA. The procedure previously describe for VICA works for these partitions too.

12.2.7.14 Procedures for VE Power Domains

VE and VE2 share the same power gate timers. The VE and VE2 partitions contain the ISP units. In practice these are only turned on when CSI or MIPI interfaces are on. These are placed in the SOR partition. Software must therefore ensure that the SOR partition is powered on when VE, and potentially VE2, are on.

VE Power Gating

Software should be aware that disabling the CSI clocks will also affect the CSICIL which is in the SOR power domain.

1. Flush MC clients VI and ISP by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.VI_FLUSH_ENABLE
   b. MC_CLIENT_HOTRESET_CTRL_0.ISP2_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:
   a. VI_HOTRESET_STATUS
   b. ISP2_HOTRESET_STATUS
3. Set the following bits to assert reset to VI, ISP and CSI:
   a. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_VI_RST
   b. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_ISP_RST
   c. CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_CSI_RST
   d. CLK_RST_CONTROLLER_RST_DEVICES_Y_0.SWR_VI_I2C_RST
4. Clear the following bits to disable clocks to VI, ISP and CSI:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_VI
   b. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISP
   c. CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
   d. CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILAB

e. CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILCD
f. CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILEF
g. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_VI_I2C

5. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = VE
   b. START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit VE is clear.

VE Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = VE
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit VE is set
3. If need, set the following bits to enable clocks to VI, ISP and CSI:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_VI
   b. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISP
   c. CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
   d. CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILAB
   e. CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILCD
   f. CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_CILEF
   g. CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.CLK_ENB_VI_I2C
4. Remove power gating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.VE
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit VE is cleared
6. If need, clear the following bits to de-assert reset to VI, ISP and CSI:
   a. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_VI_RST
   b. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_ISP_RST
   c. CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_CSI_RST
   d. CLK_RST_CONTROLLER_RST_DEVICES_Y_0.SWR_VI_I2C_RST
7. Enable MC clients VI and ISP by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.VI_FLUSH_ENABLE
   b. MC_CLIENT_HOTRESET_CTRL_0.ISP2_FLUSH_ENABLE

12.2.7.15 Procedures for VE2 Power Domain

VE and VE2 share the same power gate timers. The VE and VE2 partitions contain the ISP units. In practice these are only turned on when CSI or MIPI interfaces are on. These are placed in the SOR partition. Software must therefore ensure that the SOR partition is powered on when VE, and potentially VE2, are on.

VE2 Power Gating

1. Flush MC clients ISP2 by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.ISP2B_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:

a. ISP2B_HOTRESET_STATUS

3. Set the following bits to assert reset to ISP2:
   a. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_ISPB_RST
4. Clear the following bits to disable clocks to ISP2:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISPB
5. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = VE2
   b. START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit VE2 is clear.

VE2 Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = VE2
   b. START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit VE2 is set
3. If need, set the following bits to enable clocks to ISP2:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISPB
4. Remove power gating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.VE2
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD until bit VE2 is cleared
6. If need, clear the following bits to de-assert reset to ISP2:
   a. CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_ISPB_RST
7. Enable MC clients VI and ISP by clearing the following bits:
   a. a.MC_CLIENT_HOTRESET_CTRL_0.ISPB_FLUSH_ENABLE

12.2.7.16 Procedures for DFD Power Domain

DFD is a new power gate-able partition in Tegra X1 onwards. The power gating of DFD needs to be under the control of customer using a build time configurable switch in the Boot Loader. i.e. the code to power gate the DFD partition will be present but under a “#ifdef POWERGATE_DEBUG”. The intention of power gating DFD partition is to get continuous power savings once the chips are in end products. Later for any field failure parts that need to be debugged, the boot loader needs to be built with another option to power ungate the DFD partition. So essentially the power gate and ungate is only valid during cold boot for DFD partition. In other words, do not power down the DFD partition with the other partitions before entering LP1 in the chips where the boot loader keeps the DFD powered up. This is because customers would want to use CoreSight for debug during LP1 to access partitions that aren’t power gated.

The only time and method by which the DFD partition should be PG is via the BL under the control of a build time parameter.

DFD contains 2 independent units: coresight_soc400 and internal logic analyzer. Both need to be reset and clock gated before power gating.

Another key thing to note for DFD is that it has 2 MC interfaces (ETR and AXIAP) which need to be flushed before initiating power gating.

DFD Power Gating

1. Flush MC clients by setting the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_1_0.ETR_FLUSH_ENABLE

b. MC_CLIENT_HOTRESET_CTRL_1_0.AXIAP_FLUSH_ENABLE

2. Poll MC_CLIENT_HOTRESET_STATUS_1_0 until the following bits are set:
   a. ETR_HOTRESET_STATUS
   b. AXIAP_HOTRESET_STATUS
3. Set the following bits to assert reset to DFD
   a. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_CSITE_RST
   b. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_LA_RST
4. Clear the following bits to disable clocks to DFD:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_CSITE
   b. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_LA
5. Check APBDEV_PMC_PWRGATE_STATUS_0.DFD is set to ascertain if DFD is Powered up.
6. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = DFD
   b. START = ENABLE
7. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit DFD is clear.

DFD Power Ungating

1. Check APBDEV_PMC_PWRGATE_STATUS_0.DFD is clear to ascertain DFD is power gated.
2. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   a. PARTID = DFD
   b. START = ENABLE
3. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit DFD is set
4. Set the following bits to enable clocks to DIS:
   a. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_LA
   b. CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_CSITE
5. Remove powergating clamps by writing a 1 to the following bits:
   a. APBDEV_PMC_REMOVE_CLAMPING_CMD.DFD
6. Poll APBDEV_PMC_CLAMP_STATUS_0 until bit DFD is cleared.
7. Clear the following bits to de-assert reset to DFD:
   a. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_CSITE_RST
   b. CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_LA_RST
8. Enable MC client DFD by clearing the following bits:
   a. MC_CLIENT_HOTRESET_CTRL_0.ETR_FLUSH_ENABLE
   b. MC_CLIENT_HOTRESET_CTRL_0.AXIAP_FLUSH_ENABLE

12.2.7.17 Procedures for GPU Power Domain

Asynchronous resets arriving in GM20b are made synchronous to the osc_div_clk domain but are then consumed in multiple other clock domains. Therefore, there is a synchronization issue between the osc_div_clk and these other domains. The resets arrive during rail-gating and so compared to previous architectures, the sequence changes in bold below are required during rail-gating entry/exit.

GPU Rail-Gating

1. NV_PMC_CLIENT_HOTRESET_CTRL_1_GPU_FLUSH_ENABLE = _ENABLE
2. Wait for NV_PMC_CLIENT_HOTRESET_STATUS_1_GPU_HOTRESET_STATUS=_FLUSH_DONE
3. NV_PAPBDEV_PMC_GPU_RG_CNTRL_RAIL_CLAMP = 1
4. SWR_GPU_RST=1
5. CLK_ENB_GPU=0 and
6. **CLK_ENB_PLLG_REF=0**

GPU Rail-Ungating

1. CLK_ENB_GPU = 1 and **CLK_ENB_PLLG_REF=1**
2. NV_PAPBDEV_PMC_GPU_RG_CNTRL_RAIL_CLAMP = 1
3. NV_PAPBDEV_PMC_GPU_RG_CNTRL_RAIL_CLAMP = 0
4. SWR_GPU_RST=1
5. Delay
6. CLK_ENB_GPU=0
7. SWR_GPU_RST=0
8. CLK_ENB_GPU=1
9. NV_PMC_CLIENT_HOTRESET_CTRL_1_GPU_FLUSH_ENABLE = _DISABLE

12.2.7.18 Procedures for APE Power Domain

APE Power Gating

Tegra X1 devices support power gating of the Audio Processing Engine (APE) during scenarios when the audio subsystem is not used by software. The APE subsystem consists of an ADSP (Cortex®-A9 CPU), ADSP registers, and mailboxes, AHUB, ADMA, and other related logic. The entire cluster is one power domain referred to as APE. The reset default of the APE cluster is to be powered off.

The decision to power gate is made by the audio driver running on main CPUs. The sequence below is followed by software prior to the register writes to PMC to power gate the APE.

1. Software on CPU requests the ADSP to go idle.
2. The ADSP polls the APE clients to make sure they are IDLE.
3. The ADSP flushes caches, save context in DRAM. Sets a sticky flag to indicate context is saved.
4. The ADSP Programs GIC to route interrupts to CPU.
5. The ADSP executes WFI, which causes an interrupt to the CPU
6. This is the trigger to PG the APE.

The PMC supports power gate enable and clamp control registers for APE power gating. The CAR unit implements the reset(s) and clock enables registers.

Note: *While un-gating power to the APE, software needs to sequence the resets in accordance with the boot/PG exit requirements. For example, the ADSP reset is released only after software has written the appropriate reset vector registers.*

Refer to Chapter 23: Audio Processing Engine in this TRM for additional details.

12.2.8 MPE Power Gating

From a software perspective, MPEA and MPEB are one logical partition.

The PMC also has an option to sequence both partitions at the same time to shorten power-gating latency. The TD_PWRGATE_INTER_PART_TIMER[3:0] register can be programmed to configure the period between two partitions power-gate up/down sequence. Partitions are powered up in order – MPEA, followed by MPEB. Power gating down is executed in reverse order. Programming a 0 value in the TD_PWRGATE_INTER_PART_TIMER register will trigger simultaneous power gating of both partitions.

Note: *There is a power domain (MPE) made up of two partitions: MPEA (NVENC Partition A) and MPEB (NVENC Partition B). See Section 12.4.6: SoC Rail Partitions for details.*

12.3 Generic Timer's System Counter (TSC)

The ARM Generic Timer Specification requires a system counter (also known as a Time Stamp Counter) to be implemented in SoCs. The Cortex-A57 timers use this system counter as their reference clock input.

For the Generic Timer Specification, refer to the "The Generic Timer" section in the ARM Architectural Reference Manual (ARMv7-A).

The ARM system counter requirements can be grouped into two groups: basic counter functionality and architectural control and status registers functionality.

12.3.1 Basic Counter Functionality

The basic counter functionality requirements are:

- It should run at a constant clock frequency regardless of the power and clocking state of the processor cores using it.
- A lower bound on the clock frequency of the counter is in the range 1-10MHz.
- When the system is running with an inherently low clock frequency, it is acceptable that the precision of the counter is reduced.
- It must maintain the required counter accuracy, so that it does not gain or lose more than one second in a 24-hour period.
- The size of the counter is recommended to be 64 bits to avoid any roll-over issues. For a counter clock frequency less than 50 MHz, a 56-bit counter is acceptable.

12.3.1.1 Basic Functionality Implementation

Tegra X1 devices include a 56-bit free-running TSC counting at the main oscillator clock frequency. It is reset by pmc-rst (also known as main-rst). During deep sleep, the oscillator clock can be disabled. The PMC provides support for running the counter off a 32.768 kHz clock.

To save power, in the TSC implementation, only an offset counter runs in 32.768 kHz mode. All external (to PMC) visible views of the TSC (i.e., the TSC bus from the PMC, and TSC register readable value) remain frozen during this mode.

Refer to the TSC Clock Switch Programming Requirement section below for more information.

12.3.1.2 TSC Clock Switch Programming Requirement

Normally, the TSC runs at the oscillator clock, which may be optionally disabled during deep sleep.

Oscillator to 32.768 kHz Switch

If the oscillator needs to be disabled, then software has to ensure that the TSC clock is switched to 32.768 kHz before the oscillator is disabled. The clock switch (to 32.768 kHz) is triggered when software sets the

PMC_DPD_ENABLE[TSC_MULT_EN] bit. Hardware waits for the first rising edge (on 32.768 kHz) before switching the clock. Software needs to use the following steps to ensure the TSC clock is switched (to 32.768 kHz):

- Set the APBDEV_PMC_DPD_ENABLE_0[TSC_MULT_EN] bit
- Poll for APBDEV_PMC_TSC_MULT_0[FREQ_STS] to be '1' OR wait for 31 µs (=1 cycle of 32.768 kHz)
- OSC disable can now be initiated as part of the LP0 entry process in the PMC hardware. Software cannot disable OSC.

32.768 kHz to Oscillator Switch

Similar to the oscillator-to-32.768 kHz switch, hardware waits for the first rising edge on 32.768 kHz, before switching to the oscillator clock. It must be ensured that OSC is enabled before the TSC clock is switched to the oscillator. The PMC hardware ensures that the oscillator is enabled at the LP0 exit. The LP0 restore software has to switch the TSC clock to the oscillator by clearing the APBDEV_PMC_DPD_ENABLE_0[TSC_MULT_EN] bit. After switching to the oscillator clock, software does not have to wait for APBDEV_PMC_TSC_MULT_0[FREQ_STS] to be '0' because 32.768 kHz is always running.

12.3.1.3 GPU Time Stamp

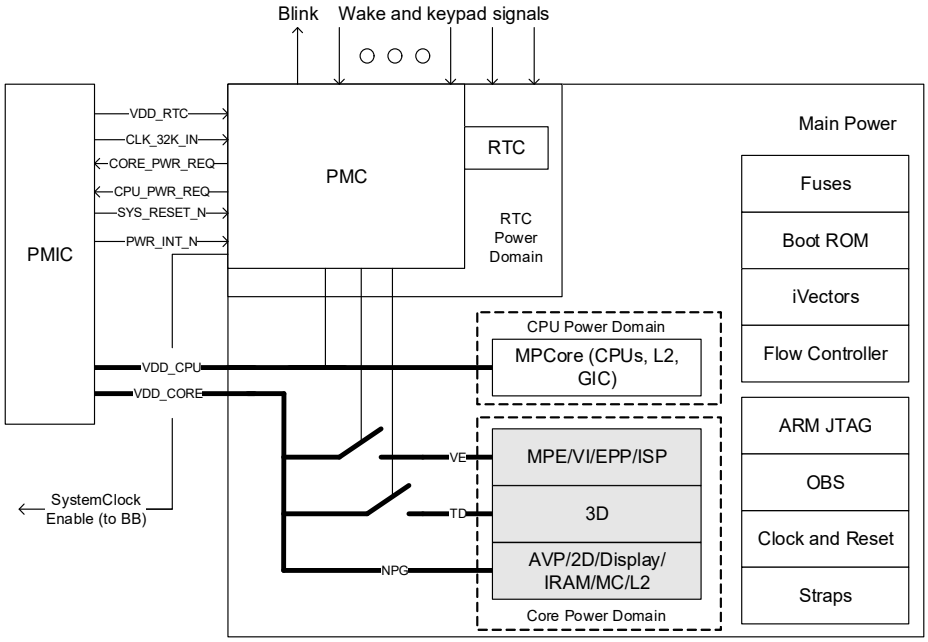
The Gray-encoded TSC is also sent to the GPU, together with a selected bit of the binary TSC acting as a tick clock to minimize power; that is, some logic in the GPU is only activated after an edge on the tick indicator.

The index of the bit selected as tick clock is programmable via APBDEV_PMC_TSC_MULT_0[TICK_SEL], which can take a value between 0 and 5.

12.4 Functionality

The PMC is a part of the RTC power domain. The PMC manages the interface with the external PMIC, including the hardware reset pin, the 32.768 kHz clock, and the power request signal. A high-level schematic view of the power domains inside the Tegra X1 devices is shown below. Refer to "SoC Rail Partitions" for a summary of all power domains.

Figure 18: High-Level View of Power Domains in Tegra X1 Devices



An important aspect of the PMC is the wake-up and reset logic. The following subsections show the expected behavior of the PMC under three scenarios:

- Frozen boot, that is, the VDD_RTC transitioning from OFF to ON, with or without VDD_CORE and VDD_CPU also transitioning from OFF to ON

- Deep Sleep wake-up, where the PMC is responsible to track wake-up events and to request a transition of VDD_CORE and optionally VDD_CPU from OFF to ON
- Suspend mode wake-up, where the PMC is responsible for establishing the power back to the CPU partition

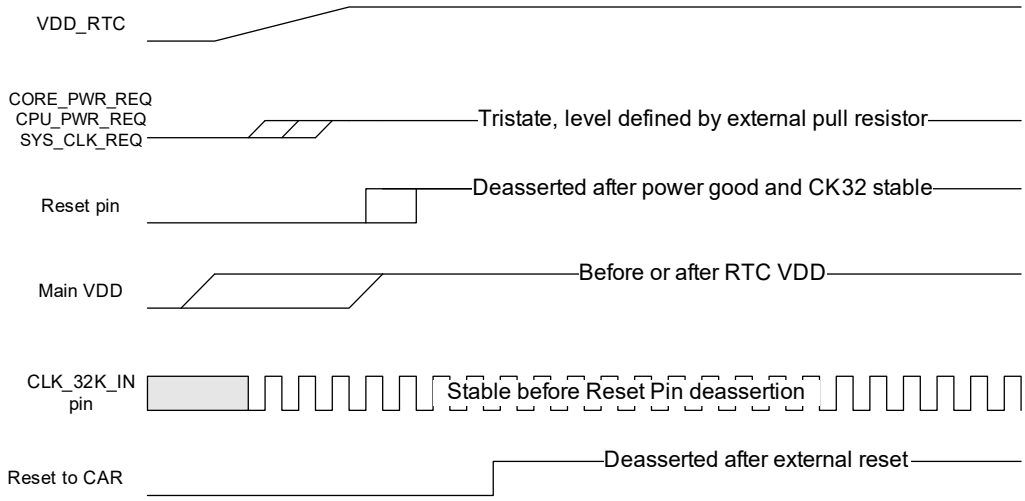
When reset is deasserted to the Tegra X1 devices, the PMC logic performs minimal processing. It forwards a reset signal and the 32.768 kHz clock to the Clock and Reset (CAR) block.

12.4.1 Frozen Boot Sequence

The Frozen Boot wake-up sequence happens when power has been deasserted to the RTC domain:

1. The PMIC detects a turn-on condition and enables power to VDD_RTC.
2. Simultaneously to VDD_RTC enable or shortly afterwards, the PMIC enables power to VDD_CORE and optionally to VDD_CPU.
3. More than 1 ms later, the PMIC enables power to 1.8V I/O rails that need to be enabled for boot.
4. More than 1 ms later, the PMIC enables power to 2.8V/3.3V I/O rails that need to be enabled for boot.
5. The PMIC drives a 32.768 kHz clock to Tegra X1 devices.
6. More than 1 ms later, the PMIC deasserts reset to Tegra X1 devices. The 32.768 kHz clock and all power rails must be stable and at their nominal value before reset is deasserted.
7. The PMC block resets and deasserts reset to the CAR block, which resets the rest of the Tegra X1 chip.
8. Tegra X1 devices begin boot from internal ROM.

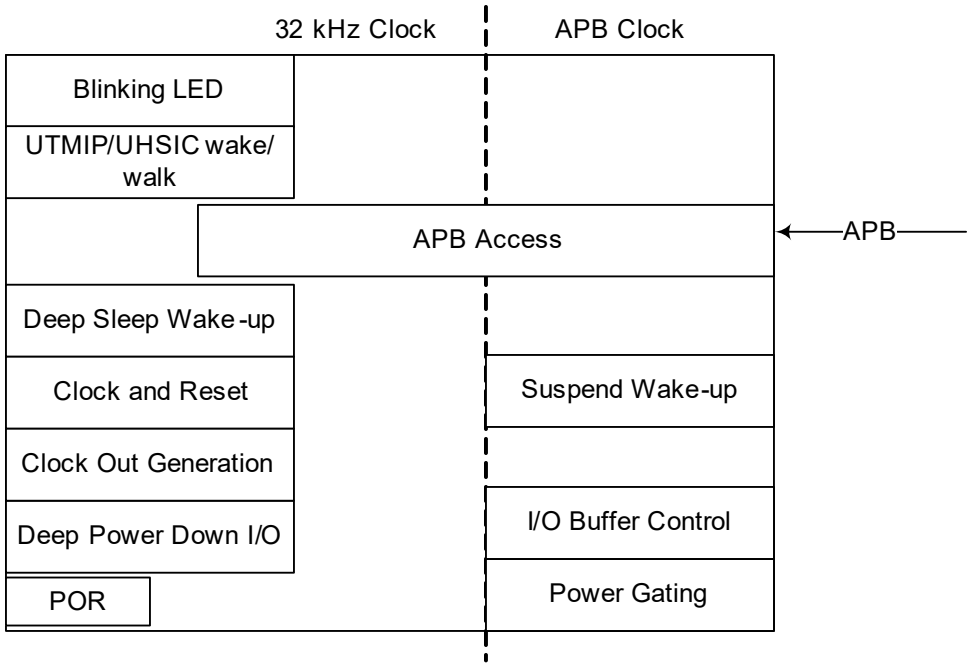
Figure 19: Frozen Boot Sequence



12.4.2 Reference Block Decomposition

The next figure shows a reference decomposition of the PMC.

Figure 20: Reference Block Decomposition Diagram



The blocks inside the PMC are:

- Wake-up logic for Deep Sleep and Suspend modes
- Power-gating logic
- External clock and reset
- Blinking LED
- I/O buffer control (software-driven I/O deep power down)
- APB interface
- UTMIP/UHSIC wake/walk logic
- Deep power down I/O logic (independent of deep sleep)
- Clock out generation

12.4.3 Always-On Thermal Alarm Generator (AOTAG)

The PMC remains powered during “standby” (LP0). It contains the only always on temperature sensor as well as associated temperature measurement and alarm condition generation logic.

The PMC-TAG temperature sensor is configured by software and always left on to monitor for low/high temperature events as well as a catastrophic thermtrip event.

Once software hot/cold events are detected, the Tegra X1 processor shall be woken up and system software shall perform appropriate system management activities for the corresponding alarm.

Once a thermtrip event is detected, it is signaled to the PMIC via an overloaded existing interrupt line.

12.4.3.1 Alarm Types

The PMC-TAG supports the detection and reporting of three alarm conditions: low temperature alarm, high temperature alarm, and thermtrip. For consistency, all alarm conditions are specified by pairs of comparators, one for alarm condition set threshold and another for the alarm condition reset threshold, allowing hysteretic condition set/reset.

12.4.3.2 Security

Event threshold control registers have separate write locking controls for security, similarly an overall AOTAG control write-lock is provided that prevents all further AOTAG register modifications. All write lock bits are cleared only by power on reset.

Power Domains

All of PMC_AOTAG logic is powered by VDD_RTC.

12.4.4 Power Gating/Ungating

The only Suspend mode wake-up specific logic in the PMC is related to the power gating logic as described previously.

12.4.5 CPU Rail Partitions

The Tegra X1 CCPLEX supports a 4+4 configuration where there is a fast cluster consisting of 4 Cortex®-A57 CPU cores and a slower cluster consisting of 4 Cortex-A53 CPU cores. Only one cluster can be in CC0/CC3/CC4 state at a given time, and the other cluster is guaranteed to be in the CC6 state (i.e., all cores and Non-CPU power gated). The cluster switch is managed by software running on BPMP-L.

Each CPU is an independent power-gated domain. The cluster's non-CPU may be power gated as a separate domain. The flow controller would provide option for cluster L2 to be power gated.

The non-CPU cannot be power-gated unless all of the CPUs are power-gated (or inactive). Also the L2 cache needs to be flushed before power-gating the non-CPU.

CPU power-gating control is separated from SoC power-gating control because the CPU rail partitions need different inter-zone delay values (configured in PMC_PWRGATE_TIMER_* registers). There are 8 zones per CPU partition.

In addition, unlike non-CPU SoC partitions (which can only be power gated/ungated by direct PMC register writes), CPU (fast or slow) related partitions can be power gated/ungated by the flow controller requesting PMC to power gate/ungate them. The following table provides a list of CPU rail partitions.

Table 36: CPU Rail Partitions

Partition	Partition Description	Notes
CE0	Cluster0 CPU0	Normally, power-gated on/off via flow-controller interaction. Can be power-gated on/off by direct register write. Uses PMC_PWRGATE_TIMER_CE* inter-zone delays.
CE1	Cluster0 CPU1	
CE2	Cluster0 CPU2	
CE3	Cluster0 CPU3	
C0NC	Cluster0 Non-CPU	
CRAIL	CPU Rail	Normally, power-gated on/off via flow-controller interaction. Can be power-gated on/off by direct register write. On/Off delays are programmed in the PMC_CPUPWRGOOD_TIMER and PMC_CPUPWROFF_TIMER registers. These timers are used only when the platform supports a CPU power good indication.

12.4.6 SoC Rail Partitions

In SoC partitions, only CPULP and C1NC can be power-gated on/off via the flow controller. All other partitions can only be power-gated (on/off) by direct register writes. Tegra X1 PG partitions have 8 zones per partition.

Partition	Gated Power Domain	Description
NVDEC	PD_nvjpg	NVDEC part A and NVJPG
NVDEC_b	PD_nvdec	NVDEC part B
NVDEC_c	PD_nvdec	NVDEC part C
AUD	PD_aud	APE sub system

Partition	Gated Power Domain	Description
ADSP	PD_aud	Audio processor
VIC	PD_vic	VIC Logic
VICB	PD_vic	VIC Logic
VICC	PD_vic	VIC Logic
VICD	PD_vic	VIC Logic
MPEA	PD_mpe	NVENC Part A
MPEB	PD_mpe	NVENC Part B
DIS	PD_dis	Display A core logic
DISM	PD_dis	Display A memory
DISB	PD_disb	Display B core logic
DISBM	PD_disb	Display B memory
SOR	PD_sor	Display interfaces
VE	PD_ve	ISP
VE2	PD_ve2	ISP2
DFD	PD_dfd	DFD debug logic
XUSBA	PD_xusba	XUSBA Device mode
XUSBB	PD_xusbb	XUSBB Device mode
XUSBC	PD_xusbc	XUSC Device mode
PCX	PD_pcx	PCIE core logic
SAX	PD_sax	SATA

12.4.7 Default Power/Clamp/Reset Status of PG Partitions

This subsection provides the default power, clamp, and reset status of the PG partitions.

12.4.7.1 Cold Boot (Power-on Reset)

By default, all of the CCPLEX related partitions (except the C0NC) are power-gated and non-CCPLEX partitions are power-ungated. The clamps are accordingly enabled (for the power-gated domain) or disabled (for the power-ungated domain). The C0NC is power-ungated by default (to keep the CAR reset defaults consistent with the power-gating status at LP0 exit).

Refer to the PMC_PWRGATE_STATUS and PMC_CLAMP_STATUS register descriptions for the default power and clamp status, respectively, of PG partitions.

12.4.7.2 Deep Sleep Exit

At deep sleep exit (LP0), the PMC retains power-gating status of all the PG partitions. The PMC does not restore the CPU rail – if necessary, software needs to power-up the CPU rail during LP0 exit (via a PMC register write).

The CAR unit is powered off during deep sleep, so the CAR provided reset defaults at LP0 exit are the same as its cold boot defaults. At LP0 exit, the C0NC is expected to be power-ungated. Therefore, its cold boot default is power-ungated, so its reset (which is provided by the CAR block) can be kept the same for cold boot and LP0 exit.

12.4.8 GPU Rail Gating

The PMC provides GPU to SoC clamp enable which needs to be set (by software) in the APBDEV_PMC_GPU_RG_CNTRL_0_RAIL_CLAMP register bit before the GPU rail is powered off, and cleared after the GPU rail is powered off. The PMC GPU power-gating logic is not used but remains unchanged (the Tegra X1 GPU has its own ELPG controller).

The GPU rail power-up sequence is as follows:

- Software requests GPU rail power up (via PMIC I2C)
- Wait for power good (based on the software timer)
- Remove the clamp (by writing the PMC_GPU_RG_CNTRL[RAIL_CLAMP] register)
- Enable the CAR to GPU clock (it is enabled by default)
- Deassert CAR to GPU reset (by writing to the CAR register named CLK_RST_CONTROLLER_RST_DEV_X_SET_0_SET_GPU_RST)

The GPU rail power-down sequence is as follows:

- Assert CAR to GPU reset (by writing the CLK_RST_CONTROLLER_RST_DEV_X_SET_0_SET_GPU_RST register)
- Disable CAR->GPU clock (by writing the CLK_RST_CONTROLLER_CLK_ENB_X_SET_0_SET_CLK_ENB_GPU register)
- Enable the clamp (by writing the PMC_GPU_RG_CNTRL[RAIL_CLMAP] register)
- Software requests GPU rail power down (via PMIC I2C)
- Wait for power off (based on the software timer)

  This step depends on the PMIC. Some PMICs do not require power off wait; however, it is likely to be needed.

12.4.9 CPU Power States

The CCPLEX implements the following core level power states:

- C0 – Core is ACTIVE and executing.
- C1 – Core is IDLE and clock gated after having executed a WFI or WFE instruction. The core has to be in this state before it can transition to a deeper power state.
- C7 – Core is power gated.

Individual cores within the same cluster can transition freely and independently between the C0/C1/C6 power states. There is no direct transition from C1 to C7 on Tegra X1. The CPU management drivers can go from C0 to C7 if the core is hot plugged or the C7 is entered from cpuidle. The hot plug policy is controlled by which CPU governor is selected. If runnable governor is selected, the CPU state will be switched depending on the running threshold. Currently, a core enters C7 from cpuidle when the predicted idle time by the cpuidle governor is more than 1 ms. Idle time is predicted using a statistical average with various other parameters influencing the metric. We enter C1 when the predicted idle time is really small (<500 us).

The entry time between C0 and C7 is ~ 0 us, and exit time is ~ 80 us.

The CCPLEX implements the following cluster level power states. They apply to both Cortex-A53 and Cortex-A57 clusters.

- CC0 – One or more cores in the cluster are active (in C0).
- CC1 – All the cores in the cluster are idle and (self) clock gated after having executed a WFI or WFE instruction. At least one core is in C1. Other cores can be in C1 or deeper state.
- CC3 (formerly HVC) – All cores are idle (in WFI mode) and VDD_CPU has been lowered to Vmin.
- CC4 Retention – All cores are idle (in WFI mode) and VDD_CPU has been lowered to Vret.
- CC6 – All cores plus L2 are power gated. The L2 content has been flushed to memory.
- CC7 – All cores plus L2 are power gated. In addition VDD_CPU rail is OFF.

12.4.9.1 CC3/CC4 States

Tegra X1 devices support two new CPU power states: CC3 and retention CC4. This flow is triggered when the CPUs are idle, and the flow controller detects STANDBYWFI assertion from the last core standing. The PMC plays the following roles in the entry sequence:

- Receives a trigger from flow controller and disables CCPLEX TSOSCs from sensing temperature. (A sideband signal driven from the PMC (pmc2soc_therm_crail_voltage_valid) is used to invalidate the temperatures sensing in SOC therm)
- De-asserts CPU_PWR_REQ to request Vmin value from PMIC.
- Notifies the flow controller that CC3 entry is complete.

On the exit sequence from CC4/CC3, the PMC does the following actions:

- When the flow controller requests the voltage change, the PMC asserts CPU_PWR_REQ to ramp up the voltage.
- Waits for a timer expiry to make sure the voltage has reached functional levels and re-enables TSOSCs in CCPLEX.
- Sends the ack back to the flow controller. [the previous step is non-blocking]

CPU retention mode is conditionally entered from the CC3 state, and BPMP-L SW is responsible for all of the retention specific steps. PMC provides a SW hook to clamp the CCLPLEX -> CORE interface through PMC_SET_SW_CLAMP register. BPMP-L SW will use this register to assert the VDD_CPU -> VDD_CORE clamp signal before VDD_CPU is brought down to retention levels. On exit from retention, this clamp is released after VDD_CPU is back at functional levels.

The latency of entering CC1 and CC3 is quick as it is hardware sequenced:

CC1: Few clocks

CC3: < 10 µs

CC4: Entry/exist involves BPMP processor which takes more latency. The entry time is < 50 µs, and exist time is 140 µs.

PMIC Control for CC3/CC4 State Transitions

The PMC supports an optional mechanism to de-assert CPU_PWR_REQ as part of CC3 flows, with the intention of signaling the need to transition to Vmin to the PMIC, instead of going over the normal VID transfer mechanism of the PMIC (i2c/PWM). This sideband signaling can be masked by programming the PMC_CC4_HVC_CONTROL.MASK_CPUP_PWRREQ bit.

The table below summarizes the regulator control mechanism and configuration settings in the PMC and CL-DVFS for the supported PMIC types.

PMIC	VID Protocol	Vmin Selection Mechanism	CL-DVFS Settings	PMC Settings	Standby_ GPIO
OVR	PWM-VID	PWM VID Transfer	DVFS_CC4_HVC.CC4_HVC_FORCE_VALUE=VID (Vmin) DVFS_CC4_HVC.CC4_HVC_FORCE_ENABLE=1 DVFS_DFLL_OUTPUT_CFG.DFLL_OUTPUT_CONFIG_DELTA_EN=0	PMC_CC4_HVC_CONTROL.MASK_CPUPWRREQ=1	Software needs to assert this I/O to enter retention
MAX8973	I2C	DVS	DVFS_CC4_HVC.CC4_HVC_FORCE_VALUE=VID (Vmin) DVFS_CC4_HVC.CC4_HVC_FORCE_ENABLE=1	PMC_CC4_HVC_CONTROL.MASK_CPUPWRREQ=0	Not used

12.4.10 Wake Events

The PMC provides support for 64 wake events and ties off unused wake-event inputs.

The wake-up logic performs the following tasks:

- Tracks the state of a programmable set of wake-up events
- Detects deep sleep wake-up condition
- Optionally generates an interrupt based on wake-up events

A wake-up event is an assertion or a change of level on any event in a set of specified pins/events, *or* an assertion of an interrupt in a set if specified interrupts (e.g., RTC interrupts). The logic that tracks for change of levels in external pins contains

a glitch filter that can be optionally disabled. Internal events/interrupts do not require de-glitching logic, but are passed through the same logic, as this is also the synchronization logic. A conceptual schematic is shown in the following figure.

The latched wake-up events are readable by software once the CPU (or CPU) is active. The logic accumulates all events that take place while in LP0 mode. Normally only one enabled event is present in the set, but simultaneous events could take place. Each bit of the latched wake-up event register is cleared by writing a '1' to it.

The wake-up detection logic can also be configured to generate WakeInterrupt in which case it accumulates events when interrupt-generation-mode is enabled.

A second set of latches is also present, with the same structure as the wake-up event register but under software control. The most important difference between the two sets is that events that occur outside of the deep sleep (LP0) mode will be captured in the second set, including:

- Events that could occur during the LP0 mode transition itself, i.e., before the hardware enters the deep sleep (LP0) state, but after software has instructed hardware to enter the LP0 state.
- Events that occur after the hardware wake-up, but before software has enabled the peripheral functions that process the signals tied to the wake-up events, i.e., the GPIO event logic.

The table below shows the wake assignments. The assignment of the external wake events is given by the specific platform's pinmux table, and this is where the system designer describes which pins the platform should wake from. This assignment is application and platform specific.

Table 37: Wake Assignments

Wake Event#	PMC Wake1 Register Bit	PMC Wake2 Register Bit	Wake signal	Internal/ External
wake0	0		Wake signal	EXT
wake1	1		Wake signal	EXT
wake2	2		Wake signal	EXT
wake3	3		Wake signal	EXT
wake4	4		Wake signal	EXT
wake5	5		Wake signal	EXT
wake6	6		Wake signal	EXT
wake7	7		Wake signal	EXT
wake8	8		Wake signal	EXT
wake9	9		aotag2pmc_wake_event	INT
wake10	10		Wake signal	EXT
wake11	11		Wake signal	EXT
wake12	12		Wake signal	EXT
wake13	13		Wake signal	EXT
wake14	14		Wake signal	EXT
wake15	15		Wake signal	EXT
wake16	16		rtc_irq	INT
wake17	17		Wake signal	EXT
wake18	18		Wake signal	EXT
wake19	19		Wake signal	EXT
wake20	20		Wake signal	EXT
wake21	21		Wake signal	EXT
wake22	22		Wake signal	EXT
wake23	23		Wake signal	EXT

Wake Event#	PMC Wake1 Register Bit	PMC Wake2 Register Bit	Wake signal	Internal/ External
wake24	24		Wake signal	EXT
wake25	25		Wake signal	EXT
wake26	26		Wake signal	EXT
wake27	27		Wake signal	EXT
wake28	28		Wake signal	EXT
wake29	29			Reserved
wake30	30			Reserved
wake31	31			Reserved
wake32		0	Wake signal	EXT
wake33		1	Wake signal	EXT
wake34		2	Wake signal	EXT
wake35		3	Wake signal	EXT
wake36		4	Wake signal	EXT
wake37		5		Reserved
wake38		6		Reserved
wake39		7	utmip_0_wake	INT
wake40		8	utmip_1_wake	INT
wake41		9	utmip_2_wake	INT
wake42		10	utmip_3_wake	INT
wake43		11	uhsic_wake	INT
wake44		12	wake2pmc_xusb_system_wakeup	INT
wake45		13	Wake signal	EXT
wake46		14	Wake signal	EXT
wake47		15	Wake signal	EXT
wake48		16	Wake signal	EXT
wake49		17	Wake signal	EXT
wake50		18	Wake signal	EXT
wake51		19	Wake signal	EXT
wake52		20	Wake signal	EXT
wake53		21	Wake signal	EXT
wake54		22	Wake signal	EXT
wake55		23	Wake signal	EXT
wake56		24	Wake signal	EXT
wake57		25	Wake signal	EXT
wake58		26	Wake signal	EXT
wake59		27	Wake signal	EXT
wake60		28	Wake signal	EXT
wake61		29	Wake signal	EXT
wake62		30	Wake signal	EXT
wake63		31	Wake signal	EXT

12.4.10.1 Deep Sleep Wake Events as Interrupts

The deep sleep wake detection logic can be used to generate an interrupt (WakeInterrupt) in active states (by configuring WAKE_DET_EN). It can also be used to detect (by configuring the WAKE_DET_EN bit) and generate (by configuring the WAKE_INT_EN bit) interrupt (WakeInterrupt) outside of the LP0 state, based on the wake event.

12.4.10.2 Clearing Wake Status

The wake status accumulated in PMC_WAKE*_STATUS registers can be cleared by software by writing the corresponding bit of the PMC_WAKE*_STATUS register to '1'. These wake status registers are also cleared by the PMC during LP0 entry provided WAKE_DET_EN is not set. If WAKE_DET_EN is set, then these registers can only be cleared by software.

The wake status accumulated in PMC_SW_WAKE*_STATUS registers can be cleared by software by writing '1' to corresponding register.

12.4.10.3 USB Wake Events

USB wakes events are expected to be configured (by setting PMC_WAKE2_LVL[ALLOW_PULSE_WAKE]=1) as pulse events. From a PMC hardware perspective, they can be configured to be pulse or level – the PMC will sequence LP0 exit in either case. However, at LP0 exit, the PMC_WAKE*_STATUS register provides the status of USB wakes only if they were configured as pulse. If they are configured as level, then the PMC_WAKE*_STATUS register will not provide status of USB wake events. The PMC_SW_WAKE*_STATUS registers provide status irrespective of pulse or level configuration.

In Tegra X1 devices, USB wake events (VBUS and USBID) are supported through the PMIC.

12.4.10.4 Wake Event Debouncing Support

Tegra X1 devices support wake event debouncing. The PMC wake event de-bounce logic runs on 1 kHz clock (derived from 32 kHz clock) and uses the same de-bounce structure as used in GPIO blocks as described below.

Functionality of Debounce Block:

- When a key is pressed on a mechanical switch, the input is passed via a synchronizer.
- The output of the synchronizer is registered to new_value (i).

The input to the debounce logic is the new_value(i), is assigned to the output only if the new_value remains stable/No signal transitions for the registered space of time i.e., defined by the register field de-bounce count in terms of the 1ms period.

PMC will support de-bouncing capability to a select set of 8 wake inputs (wake events [31:24]). The following registers will contain the configuration to enable this feature.

Refer to the DEBOUNCE_DELAY and WAKE_DEBOUNCE_EN fields in the PMC_WAKE_DEBOUNCE_EN_0 register for more information.

12.4.10.5 PMC_WAKE_DELAY

PMC_WAKE_DELAY is an existing mechanism in PMC (i.e not new in Tegra X1) to delay the servicing of wake events once CORE_PWR_REQ is turned off in LP0 entry. The settings of APBDEV_PMC_WAKE_DELAY register define the wake delay in terms of 32 KHz clock cycles. This is a configurable option to make sure that CORE_PWR_REQ stays de-asserted for a minimum amount of time, before LP0 exit starts.

12.4.11 LPx States

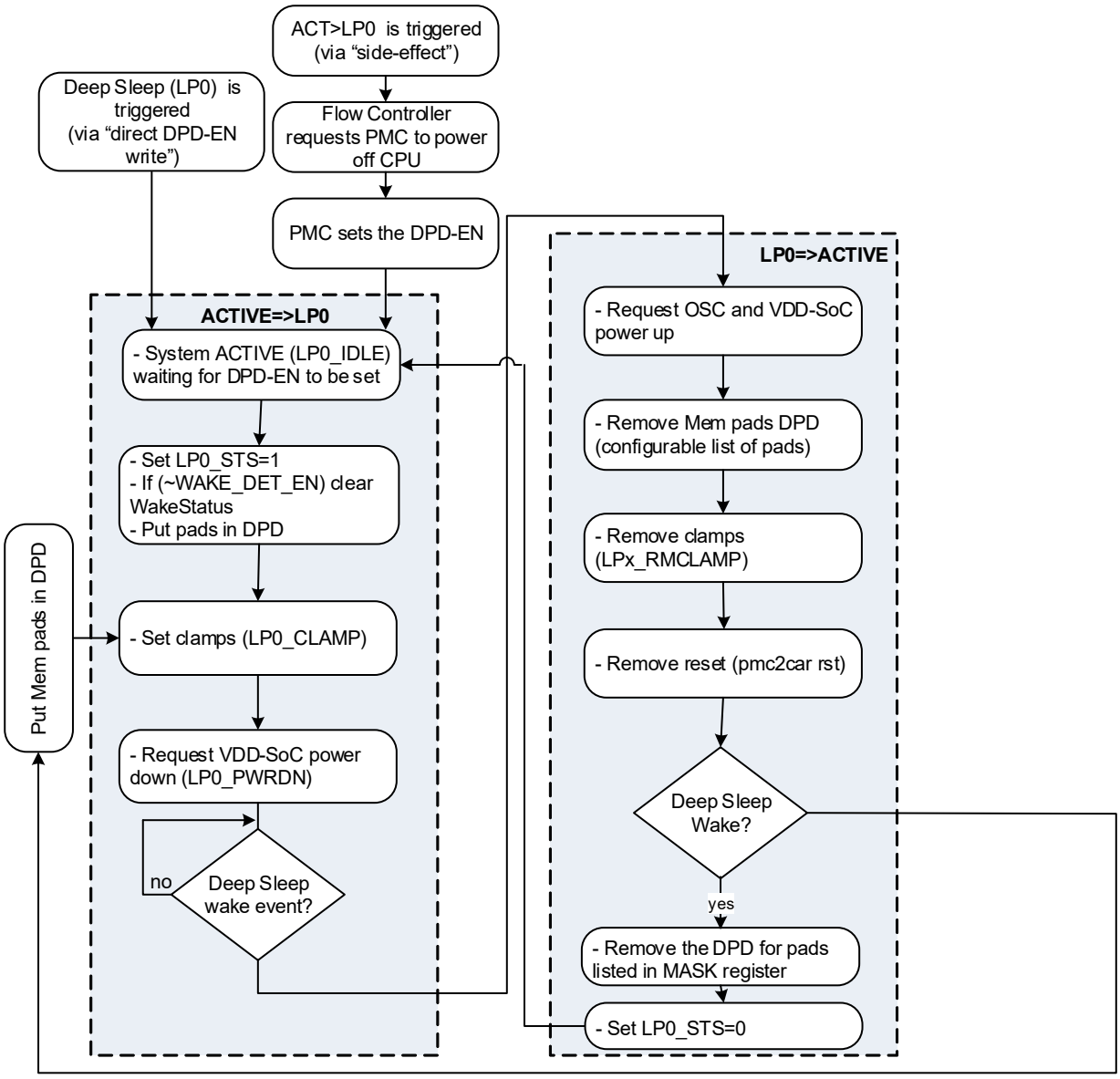
Tegra X1 processors are enabled to enter the deep sleep (LP0) state after the CPU complex enters the CC7 state. Each CPU core communicates its CC7-latency tolerance to BPMP-Lite, which then decides whether to transition the system into LP0 or not, based on the energy cost of LP0 entry and exit. LP0 is also referred to as Standby or Deep Sleep state on the platform.

The LPx states described here are system Low Power states. There are two LP states:

- LP0**:** Low Power 0 (Deep Sleep) state in which DRAM is put in self-refresh, system state is saved in PMC + DRAM, the VDD_SOC and VDD_CPU rails are powered off, and the PMC is configured to monitor “LP0 exit wake events” which would trigger LP0 exit.
- LP1**:** Low Power 1 (Suspend) state. Devices are power-gated, SoC clock domains are set to the minimum frequency (12 MHz and 38.4 MHz), the flow controller is configured to monitor “LP1 exit wake events,” DRAM is put in self-refresh, and the VDD_CPU rail is powered off.

  The following figure shows a flow diagram of the LPx states. The states marked as LP0_<> represent the states of existing LP0 state machine.

Figure 21: LPx Entry/Exit Flow Diagram



12.4.11.1 Deep Sleep Entry (ACTIVE=>LP0)

The deep sleep (LP0) entry happens after an ordered shutdown of the Tegra CPU (excluding always-on logic). As part of the shutdown procedure, most of external interfaces are placed in their idle condition (DPD – Deep Power Down). The PMC provides the logic that ensures that interfaces maintain their idle state during deep sleep. Software is responsible to sample the idle state by writing to the DPD sample bit (PMC_DPD_SAMPLE[0] bit) before entering deep sleep.

Software must also ensure that I/O paths that must remain active during LP0 are identified as such, by correct configuration of ownership bits in a PMC register if applicable.

Software must also ensure that LP0 wake-up conditions are programmed properly. Wake-up mask and wake-up level should reflect expected wake-up events. The Power Good Timer and Power Down Timer must be set according to the PMIC specification.

The deep sleep entry/wake-up logic operates largely on the 32.768 kHz clock, so software must respect a minimal delay between an LP0 exit and before issuing the next LP0 entry command. This delay is of the order of 2 periods of the 32.768 kHz clock. Normally, this should never be a problem given that the LP0 exit (Boot ROM and Recovery Code) requires much more time.

There is one way to enter deep sleep:

- *Direct-write:* Writing directly into the PMC_DPD_ENABLE register. Deep sleep is entered on POR.

The PMC deep sleep state entry is similar to existing deep sleep entry except:

- The PMC sets LP0_STS bit when it starts to enter the deep sleep state.
- The PMC clears WakeStatus only if WAKE_DET_EN=0

Refer to the PMC_CNTRL2_0 register for the LP0_STS bit description.

One time configuration:

Note: this is expected to be one (at cold boot) time configuration. However, the PMC does not care whether it is programmed once at boot or every time before deep sleep entry or never (use hardware reset default).

- [SW] Configure power-good timers. Note that the values are dependent on the PMIC specification and board configuration. PWRGOOD timers are used during LP0 exit (see “LP0 exit” for details).
  - Write PMC_PWRGOOD_TIMER register (default of 0x7f cycles of 32 kHz => 3.8 ms)
  - If the PMIC supports Power-Good signal, then enable Power-Good (See the “VDD_CPU Power-Good” section)
- [SW] Configure wake-delay. This is the minimum delay required between VDD_SOC power OFF to ON request. It is based on the PMIC specification. However, it is expected that this delay is not needed.
  - Write PMC_WAKE_DELAY register (default value 0x0)
- [SW] Setup polarity of power-request signals
  - Write PMC_CNTRL[CPUPWRREQ_POLARITY, PWRREQ_POLARITY] bits
  - Note: SYSCLK polarity is fixed (active-high)
- [SW] If need be, configure IO_DPD*_OFF_MASK registers

At every Deep Sleep entry:

- [SW] Power (and/or clock) gate units and peripherals not needed

---

Note: *PMC hardware does not care whether software power-gate non-CPU SOC partitions or not. However, it is expected that software would power-gate non-CPU SOC partitions before initiating LP0 (or LP1) entry. At the deep sleep exit, the PMC would bring SOC partitions in the same state as they were in before deep sleep entry.*

- [SW] Configure wakeup events. The corresponding I/O rail for a wakeup pin needs to be powered on in order for the wakeup to function. If no wake event is configured, then only POR reset can reset the chip.
  - Write PMC_WAKE*_MASK and PMC_WAKE*_LVL registers
- [SW] Program scratch registers with the context information needed to restore from deep sleep
  - Write context data into PMC_SCRATCH* registers

- [SW] Ensure all I/O interfaces are in idle

> ***Note:*** *The PMC does not provide any "global" idle status. Software needs to check individual devices/ IOs.*

- [SW] Put DRAM into self-refresh
- [SW] Sample IO (pads) current values which would be driven during deep sleep
  - Write PMC_DPD_SAMPLE register
- [SW] Configure pads override (which would override above sampled value)
  - Write PMC_ DPD_PADS_ORIDE register
- [SW] Write PMC_DPD_ENABLE register to trigger LP0
- [PMC] Set LP0_STS register bit to '1'
- [PMC] Request for sys-clk shutdown (this is required only when none of the 3 incoming clock requests are asserted)
- [PMC] Assert PAD mux selection signal
- [PMC] After some delay (1/2 clk32 cycle) to let mux-selection propagate, put I/Os into DPD, and put all the PLLs into DPD.
- [PMC] Clamp all the signals from VDD_CPU/SOC domain into AO or I/Os.
- [PMC] Request SOC power off
  - De-assert CORE_PWR_REQ to PMIC to turn off VDD_SOC domain

12.4.11.2 Deep Sleep Exit (LP0=>ACTIVE)

The corresponding I/O rail for a given wakeup pin needs to be powered on in order for the wakeup to function. The VDD_SOC and VDD_CPU power rails are turned off during LP0. Wakeup signals are routed from the pads directly to the RTC for the wakeup sequence to be initiated.

Deep Sleep Exit Sequence

- [PMC] Detect wake-up condition and latch the condition into the PMC_WAKE_STATUS register
  - [PMC] If enabled, latch the wake-up condition(s) into PMC_SW_WAKE_STATUS register.
- [PMC] Ensure PMC_WAKE_DELAY has passed from previous wake-event handling
- [PMC] Request OSC power up by asserting the sys-clk-req signal, and start pre-SoC Power Good timer
  - Wait for 4*PMC_PWRGOOD_TIMER[23:16] cycles of 32 kHz (note: multiplier of 4)
- [PMC] Request VDD_SoC power up by asserting the core-pwr-req signal
  - Wait for PMC_PWRGOOD_TIMER[7:0] cycles of 32 kHz
- [PMC] Remove DPD mode for all I/Os
- [PMC] Remove main (VDD_SOC domain) clamps
- [PMC] Wait for post PG OSC timer
  - Wait for 4*PMC_PWRGOOD_TIMER[15:8] number of 32kHz cycles (note: multiplier of 4)
- [PMC] Set the LP0_STS register bit to '0'
  - This update can be done before or in parallel with CAR reset de-assertion. If APB-clock is off at the time of hardware LP0_STS update, then it is okay to hold update of software-readable LP0_STS register bit until APB-clock is up and running.
- [PMC] De-assert the reset to CAR.

Note: *VDD_CPU is brought up to the same state (power on or off) it was in before entering deep sleep*

12.4.12 Power-Gating Logic

The power-gating logic is a simple programmable sequencer. When the power-gating logic is triggered, it sequences the set of power-gating signals with a programmable period (measured in cycles) between the toggling of consecutive power-gating signals.

The power-gated domains are listed in Section 12.4.5: CPU Rail Partitions and Section 12.4.6: SoC Rail Partitions. For all of them, the trigger can be a register write (see the register definitions). The CPU power-gated domain has an extra trigger coming from the flow controller. The PMC and flow controller provide a full handshake, with the PMC acknowledging the flow controller when the CPU power has completed its transition (this would normally only be for OFF to ON transition at Suspend mode wake-up).

The power-gating logic operates on the APB clock.

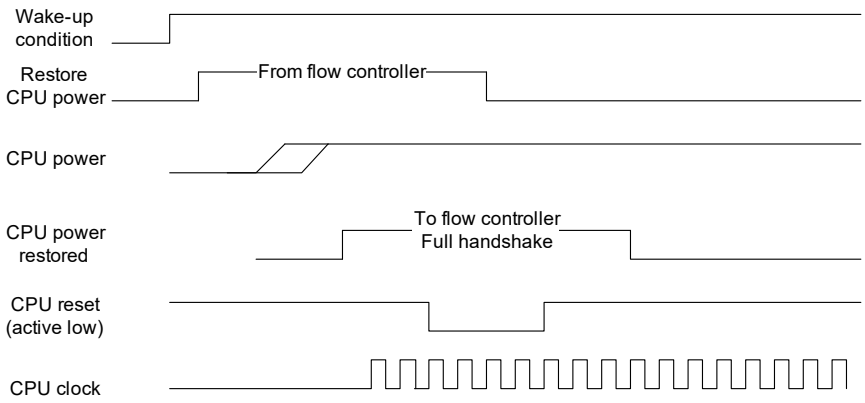
The power-gating logic may be triggered either by direct register writes using the PMC toggle register or by interaction with the flow controller. The second control path is used when entering Deep Sleep or Suspend mode to allow for an orderly shutdown of the processor. The sequence can be summarized in the following manner:

1. The CPU decides to enter either Deep Sleep or Suspend mode.
2. In both cases, power to the CPU must be gated off.
3. The CPU does any required clean-up. This is especially important for Deep Sleep (all interfaces in idle, save context, etc.).
4. The CPU informs the flow controller it wants to remove its power with two variants:
   - For Suspend mode, the CPU also defines the set of flow controller events that will wake up the CPU (the flow controller is not power gated together with the CPU).
   - For Deep Sleep, a write to a DPD sample register preserves the I/O state to be driven when the chip enters Deep Sleep mode. For Deep Sleep, the CPU must define in the PMC the set of wake-up events for the whole Tegra X1 device and set the DPD trigger enable bit.
5. The CPU enters an endless loop, executing the WFI/WFE instruction.
6. The flow controller asserts reset to the CPU.
7. The flow controller informs the PMC that CPU power-gating OFF is requested.
8. The PMC power gates the CPU OFF, then informs the flow controller.

The Deep Sleep wake-up was described before and is essentially a full reset of the core logic. The Suspend mode wake-up logic includes an interlock with the flow controller that works like this:

1. The flow controller detects the Suspend mode wake-up condition.
2. The flow controller informs the PMC that power-gating ON is requested.
3. The PMC power gates the CPU ON, then informs the flow controller.
4. The flow controller deasserts the reset to the CPU.
5. The CPU checks different status registers to establish this was in fact a Suspend mode wake-up.

Figure 22: Power-Gating Logic



12.4.12.1 Software and Hardware PG Request Interlocking

The PMC power-gating controller can only power gate (or ungate) one partition at a time. However, the PMC can get independent PG requests from software by a direct register write and hardware through the flow controller.

In Tegra 3 devices, the PMC drops the PG request when it is busy sequencing a hardware-initiated request. In Tegra X1 devices, the behavior of the PMC_PWRGATE_TOGGLE[START] register bit changed as follows:

- Software sets this bit to '1' to request a power-gate toggle.
- Hardware clears this bit to '0' when hardware starts to execute a power-gate toggle request.

For backward-compatibility, the PMC_PWRGATE_TOGGLE[START] register bit can be written to '0' by software. Normally, software should never write this bit to '0'. If software writes it to '1' and then writes it to '0' before hardware starts to execute the request, the software request gets dropped. To know the request completion, software has to poll the PMC_PWRGATE_STATUS register (where the status bit gets updated when the corresponding unit is power gated/ungated).

12.4.12.2 MPE Power Gating

In Tegra X1 devices, the MPE has two partitions that are seen as one Engine Level PG (ELPG) by software, but should be brought up/down sequentially due to power spike concerns. From a software perspective, MPE is one logical partition.

The PMC has an option to sequence both MPE partitions at the same time to shorten power-gating latency. The TD_PWRGATE_INTER_PART_TIMER[3:0] register can be programmed to configure the period between two partitions power gate up/down sequence. Partitions will be powered up in order – MPEA, followed by MPEB. Power gating down will be executed in reverse order. Programming a 0 value in the TD_PWRGATE_INTER_PART_TIMER register will trigger simultaneous power gating of both partitions.

Note: *There is a power domain (MPE) made up of two partitions: MPEA (NVENC Partition A) and MPEB (NVENC Partition B). See Section 12.4.6: SoC Rail Partitions for details.*

12.4.13 I/O Buffer Control

Deep Power Down I/O buffer management ensures that the I/Os are maintained in an idle state while in Deep Sleep mode. The procedure has been described before and requires software sequencing. See Chapter 9: Multi-Purpose I/O Pins and Pin Multiplexing (Pinmuxing) for more information.

1. Software makes sure that the affected I/O ports are in their idle state, i.e., the state that will persist during Deep Sleep.
2. When software sets the PMC_DPD_SAMPLE register, the idle state of the I/O control signals is captured, i.e., the driving direction and the data driven if the direction indicates output.
3. Software writes the DPD enable bit of the respective I/O group (PMC_DPD_REQ, PMC_DPD2_REQ, and PMC_DPD3_REQ) and muxes inside the I/O buffers now drive the stored idle state, while the corresponding control

signals coming from the core are ignored. Software can read back the PMC_DPD_STATUS and PMC_DPD2_STATUS registers to make sure that the I/O group has been placed in DPD mode.

4. Software triggers the Deep Sleep mode, killing itself.
5. Hardware detects the Deep Sleep wake-up condition.
6. Software wakes up and restores the state by writing to APB_MISC_GP* registers.
7. Software clears the APBDEV_PMC_DPD_SAMPLE sample bit.
8. I/O control can be sequenced back to the controlling block once software has initialized the core block to drive outputs to the same idle state they maintained before entering DPD.

The PMC contains software programmable registers (PMC_DPD_REQ, PMC_DPD2_REQ and PMC_DPD3_REQ), with one bit per I/O group. Each bit disables internal logic inside the I/O buffer that would otherwise consume power when the I/O power is OFF. Each bit should be set by software before shutting off the corresponding I/O power. Conversely, each bit should be reset after transitioning the I/O voltage to ON (and before enabling the set of interfaces using the corresponding I/O rail).

The PMC contains two registers related to 3.3V I/O support. Specific power detector cells per I/O voltage domain indicate if their supply voltage is low or high. The detector cells consume significant power, so they should not be left on all the time. The PMC contains a register enabling the detector cells.

12.4.14 Power Detect (PWR_DET) Controls

The reset default values of PWR_DET_VAL register will match boot rail requirements, so Boot ROM is not required to do any register updates for this functionality.

For Rails brought up by software components later in the boot flow, following steps are required to be done:

- PWR_DET_VAL bit must be set to '1' before the rail is programmed to 3.3 V
- PWR_DET_VAL bit must be cleared to '0' after rail is programmed to be at 1.8 V
- PWR_DET register bit should be ‘1’ (default is ‘1’) in order for SW to be able to update the corresponding PWR_DET_VAL register bit

---

Note: *It is illegal from pad electrical point of view, to have the rail at 3.3V without corresponding PWR_DET_VAL bit being 1.*

PWR_DET_VAL for any IO rail capable of 1.8V/3.3V operation, and are required to be used in the BR phase (e.g., IO rail for QSPI), will come up in a state to enable correct functional operation, through the detection mechanism in hardware.

Note: *Defaulting the PWR_DET_VAL at power up to indicate 3.3V operation, is electrically safe for pads capable of 3.3V/1.8V operation. However to exercise IO functionality, PWR_DET_VAL values need to match the actual rail scenario (i.e. it should indicate 1.8V if the pad is actually powered by 1.8V).*

Automotive use cases can potentially use SPI x4 as a boot media. This interface requires the PWR_DET support because the voltage could be 3.3V/1.8V at boot. This functionality is supported through the PWR_DET functionality in the SDMMC calibration pad which shares the same power rail as quad SPI.

The specified sequence to enable power detect from pads perspective is as follows:

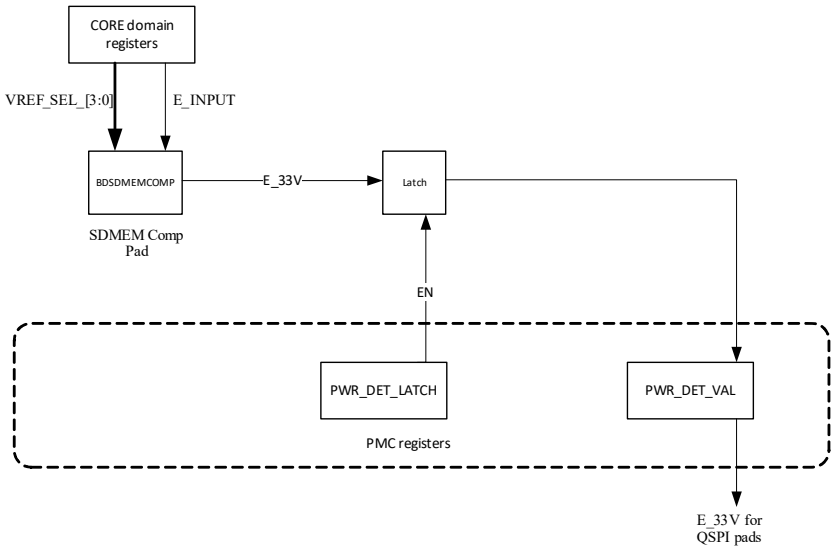
1. Set E_input=1.
   a. Make sure PWR_DET [SPI_HV] bit is ‘1’, default is ‘1’.
   b. Make sure PWR_DET_LATCH register is set to ‘1’, default is ‘1’.
2. Sample e3v3 output and provide to all pads in that interface through PMC.
   a. PWR_DET_VAL register latches the value of e3v3 and uses it to inform other pads about the rail status.

b. PWR_DET_LATCH register should be cleared, after PMC latches the value (~3usec delay), to stop PMC from latching further.

3. Disable pwrdet by setting e_input=0.

The following diagram shows the way in which PMC supports PWR DET functionality from BDSMEMCOMP pad.

The reset default of E_33V for SDMMC pad will be HIGH in the PWR_DET_VAL register. The hardware defaults of the enabling signals for the power detect functionality are expected to be in the required state to enable the propagation of E_33V to the PWR_DET_VAL register in the PMC.



The table below lists the rails that require power detect cell controls. The following rails support 3.3V operation in Tegra X1. They are shown with their corresponding register bits in the PMC_PWR_DET_VAL register.

Table 38: PMC_PWR_DET_VAL Register

Field	Description
VDDIO_SPI_HV	PWR_DET_VAL[23]
VDDIO_GPIO	PWR_DET_VAL[21]
VDDIO_AUDIO_HV	PWR_DET_VAL[18]
VDDIO_SDMMC3	PWR_DET_VAL[13]
VDDIO_SDMMC1	PWR_DET_VAL[12]

Notes:

- *The software can write to the PWR_DET_VAL register with the desired values. Individual bits of the APBDEV_PMC_PWR_DET register act as write enables for the corresponding bits of the PWR_DET_VAL register.*
- *All APBDEV_PMC_PWR_DET bits come up with the reset default of HIGH, so that the software does not need to update this register when the PWR_DET_VAL register is being configured at boot time.*

12.4.15 Power Down I/O Logic

The Tegra X1 PMC provides support for placing interfaces into a deep power down mode outside the Deep Sleep state. The following table lists the DPD domain groups:

Domain Group	Description
HDMI	Puts hdmi_pad in/out of deep power down mode

Domain Group	Description
GPIO	Puts GPIO in/out of deep power down mode
HSIC	Puts HSIC rail in/out of deep power down mode
USB3	Puts USB3 pads in/out of deep power down mode
AUDIO	Puts audio rail in/out of deep power down mode
UART	Puts UART rail in/out of deep power down mode
USB_BIAS	Puts usb_bias in/out of deep power down mode
USB0	Puts USB0 in/out of deep power down mode
USB1	Puts USB1 in/out of deep power down mode
USB2	Puts USB2 in/out of deep power down mode
PEX_CLK2	PEX clk2 pad-DPD control
PEX_CLK1	PEX clk1 pad-DPD control
PEX_BIAS	PEX bias pad-DPD control
MIPI_BIAS	Puts mipi_bias in/out of Deep Power Down mode
DSI	Puts DSI in/out of Deep Power Down mode (Named DSIA in pinout specification)
CSIB	Puts CSIB in/out of Deep Power Down mode
CSIA	Puts CSIA in/out of Deep Power Down mode
AUDIO_HV	Puts audio_hv in/out of deep power down mode
eDP	For eDP display control (APBDEV_PMC_IO_DPD2_REQ_0 [25])
DP	Puts DP in/out of deep power down mode
DMIC	Puts DMIC in/out of deep power down mode
SPI_HV	Puts SPI_HV in/out of deep power down mode
SPI	Puts SPI in/out of deep power down mode
CSIF	Puts CSIF in/out of deep power down mode
CSIE	Puts CSIE in/out of deep power down mode
CSID	Puts CSID in/out of deep power down mode
CSIC	Puts CSIC in/out of deep power down mode
DSID	Puts DSID in/out of deep power down mode
DSIC	Puts DSIC in/out of deep power down mode
DSIB	Puts DSIB in/out of deep power down mode
EMMC2	Puts EMMC2 in/out of deep power down mode
CAM	Puts CAM in/out of deep power down mode
EMMC	Puts EMMC in/out of deep power down mode
SDMMC3	Puts SDMMC3 in/out of deep power down mode
SDMMC1	Puts SDMMC1 in/out of deep power down mode

The logic for Deep Power Down ensures that the I/Os are maintained in an idle state while in Deep Sleep (LP0) mode. This requires the following SW sequencing.

Entering DPD Mode

- [SW] Set PMC_SEL_DPD_TIM to value corresponding to minimum 80 ns (this allows for proper timing sequence between sel_dpd, e_dpd)
  - This is a one time configuration, and can be done at cold boot only.
- [SW] Make sure that the affected I/O ports are in their idle state
- [SW] Set the DPD sample register (PMC_DPD_SAMPLE)
  - When software sets this register, the idle state of the I/O control signals is captured, i.e., the driving direction and the data driven if the direction indicates output.

- [PMC or SW] DPD is enabled (by direct register write)
  - The muxes inside the I/O buffers now drive the stored idle state, while the corresponding control signals coming from the core are ignored.

Note: *PMC_IO_DPDx_STATUS registers are updated to match the requests in the PMC_IO_DPDx_REQ register.*

Exiting DPD Mode

- [PMC] Hardware detects the Deep Sleep (LP0) wake-up condition
- [PMC] If configured, the PMC removes DPD enable for I/Os (see the “LP0 exit” section for details)
- [SW] Remove the DPD enable of remaining I/Os
- [SW] Clears the DPD sample bit
- [SW] Clears the DPD enable bit
  - I/O control can be sequenced back to the controlling block once software has initialized the core block to drive outputs to the same idle state they maintained before entering DPD.

Notes:

- *DPD enable is cleared by software only during LP0 > ACTIVE transitions. It has to stay asserted during deep sleep states.*
- *PMC_IO_DPDx_STATUS registers are updated to match the requests in the PMC_IO_DPDx_REQ register.*

The PMC also contains a software programmable register (PMC_IO_DPD*_REQ), with one bit per I/O “group”. Each bit disables internal logic inside the I/O buffer that would otherwise consume power when the I/O power is OFF. Each bit should be set by software before shutting off the corresponding I/O power. Conversely, each bit should be reset after transitioning back the I/O voltage to ON (and before enabling the set of interfaces using the corresponding I/O rail).

DPD mode for an I/O port can be entered independent of deep sleep state; however, in the LP0 state, the PMC automatically enables DPD mode. For regular LP0 mode, all pads will enter/exit DPD at the same time.

The following DPD domains do not have mini pad macros or do not drive state out during deep power down mode:

- MIPI_BIAS
- PEX_BIAS
- PEX_CLK1
- PEX_CLK2
- COMP
- POP_VTTGEN
- DISC_VTTGEN
- USB0
- USB1
- USB2
- USB_BIAS
- HSIC
- HDMI

- DDR_ADDR_CMD
- DISC_ADDR_CMD
- DDR_DATA
- POP_CLK

For the above domains, the sequence of entering and leaving DPD mode does not require sampling.

> ***Note:*** *Only one sample signal exists to perform sampling while entering DPD mode. This means that all interfaces will be sampled at the same time. Sequence of multiple DPD groups requires keeping all interfaces (already in DPD mode) in idle.*

12.4.16 Clock Control Logic

Tegra X1 devices support enabling an external oscillator to run in Deep Sleep (LP0) mode. It is combined with the ability to output clock (CLK_OUT) which can be extended outside deep sleep. See the PMC_OSC_EDPD_OVER and PMC_CLK_OUT_CNTRL registers for register descriptions.

There are two functions embedded in this feature:

- Programmable option of keeping the oscillator ON during DPD mode
- For low-cost systems, the ability to output the clock on aud_mclk and touch_clk pins.

> ***Note:*** *AUD_MCLK, EXTPHERIPH, and TOUCH_CLK are different names for the same pins. They are interchangeable.*

Programmable option of keeping oscillator ON during DPD mode

The OSC_CTRL_SELECT bit in the PMC_OSC_EDPD_OVER register configures whether or not the PMC starts controlling the external oscillator. Setting the EN bit to 0 (disabling the oscillator) should only be used during deep power down mode with automatic enabling on leaving deep sleep (this is to avoid system lockup). Based on PMC_OSC_EDPD_OVER setup, the external oscillator can be controlled as follows:

- Completely controlled by the CAR, entering deep power down mode on deep sleep (shutting down oscillator)
- Controlled by the PMC, entering deep power down mode on LP0
- Controlled by the PMC, not entering deep power down mode, but the oscillator output is disabled (used to accelerate warm boot time)
- Controlled by the PMC, not entering deep power down mode, the oscillator output enabled – used for clock out feature.

For low-cost systems ability to output clock on aud_mclk and touch_clk pins

aud_mclk and touch_clk are the OSC clock outputs from Tegra X1. Tegra X1 processors do not support external clock request inputs.

12.4.17 DSI Graceful Exit from Deep Sleep

This section describes how to reset the DSI logic in order to drive the idle state on the DSI pads. Follow the sequences below when entering and exiting deep sleep.

The PMC_DSI_SEL_DPD register is used to enable sel_dpd control.

Entering Deep Sleep

1. DSI sequences to move DSI brick to idle state
2. Software writes to PMC to take over sel_dpd on DSI brick pad.

3. The PMC sets sel_dpd to the brick.
4. As a result, the brick continues driving the deep sleep (LP0) state.
5. When real DPD is set, the brick continues driving deep sleep state

Exiting Deep Sleep

1. The PMC continues driving deep sleep through sel_dpd
2. Reset applied to chip – DSI resets
3. After reset is removed, software programs DSI to sequence to idle (the PMC still drives the brick)
4. Software configures the PMC to remove sel_dpd

Additionally, since control might be applied to both DSI pads – dsi, and csi_b, the PMC snoops csib_mode to decide if it should control pad csi_b when the pad serves in DSI mode.

12.4.18 Resets

The PMC receives the primary chip reset (from the SYS_RESET_N_ pad) and generates various resets for itself, the RTC, the CAR, etc. From the PMC provided reset, the CAR unit generates resets for most of the units in the chip.

In addition to chip reset, the PMC receives other events (thermal, WDT, SW, deep sleep wake), which also result in variants of the system reset. The cause of reset is captured in the PMC_RST_STATUS register.

12.4.18.1 Reset due to Thermal Trigger (Thermtrip)

The PMC can be enabled (see the PMC_SENSOR_CTRL register) to monitor the thermal trigger signal which is asserted by the SOC_therm unit when temperature goes above “thermal trip”.

The status bit (PMC_RST_STATUS) will be set to indicate that thermal trigger event has happened. If the PMC is entering deep sleep while thermtrip triggers, the thermtrip reset as will not happen (to avoid destruction of PMC state). However, process of entering deep sleep will cause reset at few cycles later.

When thermal trigger is asserted, the PMC resets the *chip*, but keeps the cause of reset (in the PMC_RST_STATUS register) and retains contents of scratch registers (including scratch0). Architecturally, the reset triggered by thermal trigger results in cold boot with the PMC retaining the information (in the PMC_RST_STATUS register) about the cause of cold boot. The behavior of thermal-triggered reset is as follows:

- The PMC receives thermal trigger
- The PMC sets the status bit
- If PMC_SENSOR_CTRL[ENABLE_RST] is set, then everything is reset except the status register (assert pmc2car_reset, pmc2rtc_reset, pmc2kmc_reset)
- De-assert resets

As a result of this reset, the system reboots (more like cold boot). During this reboot, the Boot ROM reads the cause of reset, and, if it was a thermtrip caused reboot, the Boot ROM performs a *system* reset or system shutdown via the PMIC or GPIO (based on the command in the scratch registers).

12.4.18.2 Direct Shutdown on thermtrip

The PMC optionally asserts SHUTDOWN pin to the off-chip PMIC when a thermtrip alert triggers. The PMIC treats the SHUTDOWN sideband as a failsafe trigger for shutting the system down, following which the user has to manually turn the system on by pressing the power button or equivalent inputs. The SHUTDOWN pin (formerly RESET_OUT_N) supports this feature. The PMC provides a configuration bit PMC_RST_CONFIG_THERMTRIP_EN to enable the failsafe shutdown option. System software can choose to enable failsafe shutdown or continue with the Boot ROM-based shutdown solution. Setting PMC_SENSOR_CTRL_ENABLE_RST without setting THERMTRIP_EN causes the PMC to work in the legacy mode. When THERMTRIP_EN is set, the PMC asserts the internal reset when it asserts SHUTDOWN but does not de-assert the internal

reset when the source of thermtrip is cleared in hardware. The PMC continues to power-gate CPU partitions on thermtrip alert as in previous Tegra devices, regardless of direct shutdown being enabled.

SHUTDOWN is always an active low signal from the Tegra X1 perspective. This pin is driven low only when the Tegra X1 device wants to shut down the system. At all other times, this output is tri-stated. This flexibility allows other external platform components to assert the shutdown pin to the PMIC.

The SHUTDOWN pin state is not affected by any internal system reset event. The pin, when asserted, reverts to its default only when external chip reset gets asserted.

Note: *The SHUTDOWN output is always treated as an active LOW signal, and the platform is expected to enable appropriate external pull ups on this signals. If the PMIC supports an OTP option to choose the polarity of the shutdown input, it is critical that this setting is configured to enable ACTIVE LOW operation.*

The PMIC is expected to handle SHUTDOWN as a failsafe mechanism and ensure that it is not overridden by any other input to the PMIC from the system. The PMIC maintains a status of the cause for system shutdown and software would have to read the status from PMIC. At the end of a direct shutdown event, the RST_SOURCE field in the PMC_RST_STATUS register reads as POR. The PMIC requires the SHUTDOWN signal to be kept asserted for a minimum debounce period of 244 us (8 cycles of 32 KHz clock). This period is programmable in the SHDN_ASSERT_PULSE_WIDTH counter. The counter runs at 32kHz clock and gets reset only by external chip reset.

If PMC_SENSOR_CTRL_ENABLE_RST is set, Boot ROM may restart and launch PMIC commands during the shutdown process. To prevent this from occurring when direct shutdown is enabled, THERMTRIP_EN should override SENSOR_CTRL_ENABLE_RST.

12.4.18.3 Reset due to Watchdog Timer (WDT)

If the WDT is enabled (by Boot ROM via fuse bit setting), then at the expiry of the WDT, a watchdog reset is asserted. The PMC monitors this WDT reset signal.

If a WDT reset is asserted then the PMC resets chip similar to thermtrip reset (as described in the previous section), and stores the information about the cause of reset in the PMC_RST_STATUS register.

Note: *Refer to Chapter 8: Timers for the WDT programming model.*

12.4.19 Clamping

When a partition is power gated OFF, all signals leaving that partition are clamped to their idle value.

Applying clamping presents no problems, because all blocks into the partition are powered OFF and assumed to be in reset prior to the power gate transition.

Removing the clamping is more complex because Tegra X1 devices use a synchronous reset strategy, so the correct sequence is the following:

- Restore power, reset and clamping are both active at that time
- Restore clocks, so that reset conditions propagate
- Remove clamping
- Remove reset

For maximum flexibility, removing the clamping can be controlled by software. Writing to a trigger register removes the clamping for the identified set of partitions. This does not work for Suspend mode exit, where the CPU must be restarted by hardware only.

In Suspend mode exit, there is an interlock between the Clock And Reset (CAR) block and the PMC, so that the correct sequence is performed. The CAR indicates to the PMC when the CPU clamping may be removed. This signal is asserted after the clock to CPU is restarted. The reset to CPU will only be removed after a known delay following the request to remove the clamping. The PMC removes the CPU clamping as soon as it receives the request to do so. This is similar to the interlock between the PMC and flow controller, except that this is not a full handshake.

12.4.20 Scratch Registers

PMC provides a number of scratch, secure scratch and bond out registers. The scratch registers are used to save some of the system context information during LP0.

These registers are implemented as latch-array with the exception of SCRATCH_0 register. 120 of these registers can be used as secure registers in Tegra X1.

12.4.20.1 Secure Registers

Of the scratch registers, 24 are independently secure read/write registers (PMC_SECURE_SCRATCH*). From PMC perspective, these are general purpose secure read/write registers (except Secure Registers 4-7 which are used by SE).

12.4.20.2 Access Restrictions of PMC Registers

Certain registers in the PMC can be accessed only in the TrustZone® secure mode (PNS=0). Following table summarizes these registers.

Register	Description	PNS Bit Usage
DPD_ENABLE[1]	TSC multiplier enable	Only TrustZone Write is allowed when PMC_SEC_DISABLE.TSC_NS_WRITE is set.
PMC_FUSE_CONTROL[17:16], [9:8]	PS18 latch controls for KFUSE/FUSE	Only TrustZone Write is allowed when PMC_SEC_DISABLE.PS18_NS_WRITE is set.
PMC_CRYPTO_OP[0]	Crypto Engine Disable.	When in secure_boot_mode, this register can be written to any value regardless of secure mode; When not in secure_boot_mode, only TrustZone secure request can write this register to 1
CNTCR	TSC Counter Control Register	Can be accessed only when PNS =0
CNTSR	Counter Status Register	Can be accessed only when PNS =0
CNTCV0	Counter Count Value[31:0] Register	Can be written only when PNS =0
CNTCV1	Counter Count Value[63:32] Register	Can be written only when PNS =0
CNTSR	Counter Status register	Read Only. Can be read only when PNS = 0
CNTFID0	Frequency table entry register	TrustZone READ. Write Once.
CNTFID1	Frequency table end marker	TrustZone Read only.
COUNTERID[0-11]	Counter ID registers	TrustZone Read, 5 to 7 are read only. Others can be WO.

12.4.21 LED Blinking

PMC provides the control for blinking LED including in Deep Sleep mode. The blinking control is a PWM signal.

To generate a PWM signal, PMC contains a counter with two programmable values indicating the On and Off periods (in multiples of 16 cycles of the 32.768 kHz clock). The counter is 16 bits long, for a maximum ON or OFF period of 2^{16} * 16 / 32.768 10^3 or about 16 seconds.

DATA_ON/DATA_OFF calculation:

DATA_ON time: (DATA_ON<<4) * 30.51 us + 30.51 us.

DATA_OFF time: (DATA_OFF<<4) * 30.51 us + 30.51 us.

Programmed value of DATA_ON=0 in APBDEV_PMC_BLINK_TIMER_0 will result in 1 Ton i.e., LED output will be high for at least 1 clock cycle. So the range for DATA_ON is 30.51 us to 16 seconds.

The programmed value of DATA_OFF=0 in APBDEV_PMC_BLINK_TIMER_0 will result in 1 Toff i.e., LED output will be low for at least 1 clock cycle. So the range for DATA_OFF is 30.51 us to 32 seconds

APBDEV_PMC_BLINK_TIMER register can be programmed only if the BLINK_EN in APBDEV_PMC_CNTRL_0 register is set to 0. The following programming guideline should be followed for programming the BLINK registers.

Programming guidelines:

1. Set BLINK_EN=0 in APBDEV_PMC_CNTRL_0.
2. Wait for a minimum of 2 32KHz cycles.
3. Update values in APBDEV_PMC_BLINK_TIMER_0.
4. Wait for a minimum of 2 32KHz cycles.
5. Set BLINK_EN=1 in APBDEV_PMC_CNTRL_0.

12.4.22 PLL Controls to Accelerate WB0

The controls are added for PLL (PLLP, PPM, and PLLU) override during warm boot 0 (WB0). The PMC register (PMC_PLL{P,M}_WB0_OVERRIDE) can be configured to accelerate WB0 (warm boot) process. From a usage perspective, PLLU does not need to be powered up (by the PMC) at deep sleep.

12.4.23 Pinmuxing

Some of the signals controlled by the PMC may not be required in certain customer designs, especially:

- The system clock may not support an enable function, so the system clock enable (SYS_CLK_REQ) is not used.
- The system may not require an LED active in Deep Sleep, so the power LED is not used.

It is important to allow the corresponding balls to be reused for other purposes via pinmuxing. This is not directly a responsibility of the PMC but the correct pinmuxing must be present. Tegra X1 devices provide a set of ownership registers that control if the RTC logic (PMC) has control of pins with multiple uses (PMC or typically GPIO), some of which are not controlled by the RTC logic. The ownership bit controls a final stage of muxing on top of the more general pinmuxing stage.

12.4.24 APB Access

The APB access logic is complicated by the need to not unduly block the bus for extended periods of time even when accessing registers that are logically in the 32 kHz domain. The following principles are used to avoid any delay when accessing registers logically in the 32 kHz domain:

- Write operations always take place in the APB clock domain; i.e., the corresponding registers are placed in the APB clock domain and passed as wires to the 32 kHz domain where they are used without further synchronization. This only works because values passed in this manner are considered essentially static. There is an insignificant probability that the 32 kHz logic sees an intermediate value of a multiple bit field, but an enumeration of the affected fields shows that this cannot result in any significant problem at the system level.
- Read operations are to shadow registers in the APB clock domain. The shadow registers continuously reflect the values of the equivalent registers found in the 32 kHz clock domain. Only registers reflecting status present in the 32 kHz domain are affected.

12.5 Register Definitions for the PMC

To speed up operation, the PMC register file operates in the local peripheral interface bus domain (APB) rather than in the 32 kHz clock domain used for PMC processing.

Registers in the PMC control entering/exiting Deep Sleep/Suspend mode, power detect, and I/O power functions. They also control CORE_PWR_REQ, SYS_CLK_REQ, and LED_BLINK pins.

The following registers should be programmed/read for different power events in order for the PMC to function properly.

> ***Note:*** *The APBDEV_PMC_NO_IOPOWER register contains control bits that are used to electrically isolate in-bound paths (for example, signals entering the core voltage domain from I/O power domain) in pads, when their I/O power supply (or VDDP) is powered OFF. Before an I/O rail is powered off, this bit must be set to HIGH. Similarly after the I/O rail is powered UP, this bit needs to be cleared to the LOW state so that the pad can be used for functional purposes.*

BEFORE/ON ENTERING DEEP SLEEP MODE

- PMC Control Register (bits PWRREQ_POLARITY, PWRREQ_OE, SYSCLK_POLARITY,SYSCLK_OE, LATCHWAKE_EN)
- WAKE_MASK
- WAKE_LVL
- DPD_PADS_ORIDE (for required overrides)
- PWRGOOD_TIMER
- DPD_SAMPLE
- DPD_ENABLE

AFTER WAKE-UP FROM DEEP SLEEP

- WAKE_STATUS
- SW_WAKE_STATUS
- DPD_ENABLE
- PMC Control Register (bit LATCHWAKE_EN)

BEFORE/ON POWERING DOWN PARTITIONS

- PWRGATE_STATUS
- PWRGATE_TIMER_OFF (for power down)
- PWRGATE_TIMER_ON (for power up). This has been deprecated.
- PWRGATE_TOGGLE
- REMOVE_CLAMPING_CMD (for power up)

FOR BLINK

- BLINK_TIMER
- PMC Control Register (bit BLINK_EN)
- DPD_PADS_ORIDE (for blink field)

All other operations require single register access.

12.6 PMC Registers

Refer to Section 1.4: Reading Register Tables in the Introduction section for the register table protocol as well as recommendations for accessing registers.

12.6.1 APBDEV_PMC_CNTRL_0

This register controls the clock to the RTC, and the reset to the CAR and RTC. It also manages polarity and output enable of the sys_clk_req and core_pwr_req pins. At reset, both are disabled since the PMIC properties are unknown.

Auxiliary functions include enabling of blinking functions, side-effect option, and software wake-up latching.

The deep sleep (LP0) entry is to be triggered by the PMC side-effect option. Software needs to configure the PMC for the side-effect option before triggering the power off of last CPU. If the side-effect option is set, then the PMC would initiate deep sleep in the following cases:

- A power-off request for the CPU rail will lead to CPU rail power-off followed by LP0 entry.
- A power-gating request for cluster1 non-CPU (i.e., C1NC) will lead to power-gating of the C1NC followed by LP0 entry

PMC Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00410a00 (0bxxxxxxxxx10000010000101000000000)

Bit	Reset	Description
22	0x1	SHUTDOWN_OE: Enable shutdown pad 0 = DISABLE 1 = ENABLE
21:20	0x0	CPUPWRGOOD_SEL: CPUPWRGOOD pin select. There are 4 potential source pins (soc_therm_oc{1,2,3,4}) used as PWRGOOD. This field needs to be programmed (at boot) based on which pin is used as PWRGOOD. 0 = SOC_THERM_OC1 (default) 1 = SOC_THERM_OC2 2 = SOC_THERM_OC3 3 = SOC_THERM_OC4
19	0x0	CPUPWRGOOD_EN:CPU_PWR_GOOD (From PMIC to PMC) signal is enabled. 0 = DISABLE 1 = ENABLE
18	0x0	FUSE_OVERRIDE: Fuse override 0 = DISABLE 1 = ENABLE
17	0x0	INTR_POLARITY: Inverts INTR polarity 0 = DISABLE 1 = ENABLE
16	0x1	CPUPWRREQ_OE: Power request output enable. Resets to tri-state 0 = DISABLE 1 = ENABLE
15	0x0	CPUPWRREQ_POLARITY: Inverts power request polarity 0 = NORMAL 1 = INVERT
14	0x0	SIDE_EFFECT_LP0: When set, causes the side effect of entering deep sleep after powering down the CPU 0 = DISABLE 1 = ENABLE
13	0x0	AOINIT: AO initialized purely software diagnostic and interpretation 0 = NOTDONE 1 = DONE
12	0x0	PWRGATE_DIS: Disable power gating - global override, will override function of PWRGATE_TOGGLE register. All partitions will stay enabled. **Note**: Only write to this bit when CRAIL is on. 0 = DISABLE 1 = ENABLE
11	0x1	SYSCLK_OE: Enables output of system enable clock - works only if the SYS_CLK field in DPD_PADS_ORIDE is set to 1. 0 = DISABLE 1 = ENABLE
10	0x0	SYSCLK_POLARITY: Inverts SYS_CLK_REQ enable polarity 0 = NORMAL 1 = INVERT

Bit	Reset	Description
9	0x1	PWRREQ_OE: CORE_PWR_REQ output enable. Resets to tristate 0 = DISABLE 1 = ENABLE
8	0x0	PWRREQ_POLARITY: Inverts CORE_PWR_REQ polarity 0 = NORMAL 1 = INVERT
7	0x0	BLINK_EN: Enables blinking counter and blink output works only if the BLINK field in DPD_PADS_ORIDE is set to 1. 0 = DISABLE 1 = ENABLE
6	0x0	GLITCHDET_DIS: Reserved
5	0x0	LATCHWAKE_EN: Enables latching wakeup events - stops latching on transition from 1 to 0 (sequence - set to 1,set to 0) 0 = DISABLE 1 = ENABLE
4	0x0	MAIN_RST: Resets everything but scratch 0 and reset status. 0 = DISABLE 1 = ENABLE
3	0x0	KBC_RST: Reserved
2	0x0	RTC_RST: Software reset to RTC. 0 = DISABLE 1 = ENABLE
1	0x0	RTC_CLK_DIS: Disable 32 KHz clock to RTC. 0 = DISABLE 1 = ENABLE
0	0x0	KBC_CLK_DIS: Reserved

12.6.2 APBDEV_PMC_SEC_DISABLE_0

This register disables access (read/write to secure scratch registers). Once writes are disabled, secure registers cannot be written until power on reset or reset when exiting Deep Sleep mode.

Once reads are disabled, reads from scratch registers will return 0 until power on reset or reset when exiting Deep Sleep mode.

Single register can be disabled for reads/writes; bits 0 and 1 have an overwrite function.

Secure Register Disable

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxx00000000000000000000000)

Bit	Reset	Description
22	0x0	PS18_NS_WRITE: Non-secure (i.e., non-TrustZone) write-disable for PS18_SET/CLEAR for Fuse/Kfuse 0 = OFF 1 = ON
21	0x0	TSC_NS_WRITE: Non-secure (i.e., non-TrustZone) write-disable for the PMC_TSC_MULT register and the PMC_DPD_ENABLE[TSC_MULT_EN] register bit. 0: Non-secure or secure can write. 1: Only secure can write. This bit is write-once only. Once it is set to '1', it remains '1' until reset by deep sleep or system reset. 0 = OFF 1 = ON
20	0x0	AMAP_WRITE: Disable writes to the PMC_GLB_AMAP_CFG register 0 = OFF 1 = ON
19	0x0	READ7: Disable reads from secure register 7 0 = OFF 1 = ON

Bit	Reset	Description
18	0x0	WRITE7: Disable writes to secure register 7 0 = OFF 1 = ON
17	0x0	READ6: Disable reads from secure register 6 0 = OFF 1 = ON
16	0x0	WRITE6: Disable writes to secure register 6 0 = OFF 1 = ON
15	0x0	READ5: Disable reads from secure register 5 0 = OFF 1 = ON
14	0x0	WRITE5: Disable writes to secure register 5 0 = OFF 1 = ON
13	0x0	READ4: Disable reads from secure register 4 0 = OFF 1 = ON
12	0x0	WRITE4: Disable writes to secure register 4 0 = OFF 1 = ON
11	0x0	READ3: Disable reads from secure register 3 0 = OFF 1 = ON
10	0x0	WRITE3: Disable writes to secure register 3 0 = OFF 1 = ON
9	0x0	READ2: Disable reads from secure register 2 0 = OFF 1 = ON
8	0x0	WRITE2: Disable writes to secure register 2 0 = OFF 1 = ON
7	0x0	READ1: Disable reads from secure register 1 0 = OFF 1 = ON
6	0x0	WRITE1: Disable writes to secure register 1 0 = OFF 1 = ON
5	0x0	READ0: Disable reads from secure register 0 0 = OFF 1 = ON
4	0x0	WRITE0: Disable writes to secure register 0 0 = OFF 1 = ON
3:2	0x0	NOT_USED: legacy - redundant - not used
1	0x0	READ: Disable reads from secure registers. This global disable bit is only for secure scratch registers 0 to 23. 0 = OFF 1 = ON
0	0x0	WRITE: Disable writes to secure registers. This global disable bit is only for secure scratch registers 0 to 23. 0 = OFF 1 = ON

12.6.3 APBDEV_PMC_PMC_SWRST_0

Kept for binary code compatibility only. Does not do anything.

Register Write which Resets PMC Only

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	RST: Software reset to the PMC only 0 = DISABLE 1 = ENABLE

12.6.4 APBDEV_PMC_WAKE_MASK_0

The APBDEV_PMC_WAKE registers handle wake events whose number is less than or equal to 32. For wake events whose number is greater than 32, refer to the APBDEV_PMC_WAKE2 registers below.

Current pin assignments wake_mask, wake_status, sw_wake_status, and wake levels:

- WAKE_STATUS[0] = pex_wake_n_IB (wake event 0)
- WAKE_STATUS[1] = gpio_pa6_IB (wake event 1)
- WAKE_STATUS[2] = qspi_cs_n_IB (wake event 2)
- WAKE_STATUS[3] = spi2_mosi_IB (wake event 3)
- WAKE_STATUS[4] = gpio_pe6_IB (wake event 4)
- WAKE_STATUS[5] = gpio_pe7_IB (wake event 5)
- WAKE_STATUS[6] = uart2_cts_IB (wake event 6)
- WAKE_STATUS[7] = uart3_cts_IB (wake event 7)
- WAKE_STATUS[8] = wifi_wake_ap_IB (wake event 8)
- WAKE_STATUS[9] = aotag2pmc_wake_event (wake event 9)
- WAKE_STATUS[10] = gpio_ph6_IB (wake event 10)
- WAKE_STATUS[11] = nfc_int_IB (wake event 11)
- WAKE_STATUS[12] = gen1_i2c_sda_IB (wake event 12)
- WAKE_STATUS[13] = gen2_i2c_sda_IB (wake event 13)
- WAKE_STATUS[14] = gpio_pk4_IB (wake event 14)
- WAKE_STATUS[15] = gpio_pk6_IB (wake event 15)
- WAKE_STATUS[16] = rtc_irq (wake event 16)
- WAKE_STATUS[17] = sdmmc1_dat1_IB (wake event 17)
- WAKE_STATUS[18] = sdmmc2_dat1_IB (wake event 18)
- WAKE_STATUS[19] = hdmi_cec (wake event 19)
- WAKE_STATUS[20] = gen3_i2c_sda (wake event 20)
- WAKE_STATUS[21] = gpio_pl1_IB (wake event 21)
- WAKE_STATUS[22] = CLK32K_OUT_IB (wake event 22)
- WAKE_STATUS[23] = pwr_i2c_sda_IB (wake event 23)
- WAKE_STATUS[24] = button_power_on_IB (wake event 24)
- WAKE_STATUS[25] = button_vol_up_IB (wake event 25)
- WAKE_STATUS[26] = button_vol_down_IB (wake event 26)
- WAKE_STATUS[27] = button_slide_sw_IB (wake event 27)
- WAKE_STATUS[28] = button_home_IB (wake event 28)

This register masks which event can cause a wake-up from Deep Sleep mode. It has to be set up before entering Deep Sleep mode. It works in conjunction with the WAKE_LVL register.

Only enabled events at the proper wake_lvl will cause exit from Deep Sleep mode.

PMC Wake-up Event Mask

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EVENT_RES2: general wake 0 = DISABLE 1 = ENABLE
30	0x0	EVENT_RES1: 0 = DISABLE 1 = ENABLE
29	0x0	EVENT_RES4: 0 = DISABLE 1 = ENABLE
28:23	0x0	EVENT_RES: 0 = DISABLE 1 = ENABLE
22:19	0x0	EVENT_1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
18	0x0	EVENT_RES3: 0 = DISABLE 1 = ENABLE
17	0x0	KBC: Reserved
16	0x0	RTC: RTC wake enable 0 = DISABLE 1 = ENABLE
15:0	0x0	EVENT: pin 0-15 wake enable 0 = DISABLE 1 = ENABLE

12.6.5 APBDEV_PMC_WAKE_LVL_0

This register sets the active level for the wake event. It will cause an exit from Deep Sleep mode if the input signal level matches the level set in this register and WAKE_MASK is set for the event to 1.

PMC Wake Level

Offset: 0x10 │ Read/Write: R/W │ Reset: 0xff9fffff (0b11111111100111111111111111111111)

Bit	Reset	Description
31	0x1	EVENT_RES2: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
30	0x1	EVENT_RES1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
29	0x1	EVENT_RES4: 1 to 0 pulse should be detected as wake 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
28:23	0x3f	EVENT_RES: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

Bit	Reset	Description
22:19	0x3	EVENT_1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
18	0x1	EVENT_RES3: Power interrupt - permanently tied to bit 18 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
17	0x1	RESERVED: This bit is reserved
16	0x1	RTC: RTC wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
15:0	0xffff	EVENT: Pin 0-15 wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

12.6.6 APBDEV_PM

This register stores the status of the wake events. The event will be set if the level matches and is not masked.

A write will reset the set wake event.

PMC Wake Status

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EVENT_RES2: 0 = NOT_SET 1 = SET
30	0x0	EVENT_RES1: 0 = NOT_SET 1 = SET
29	0x0	EVENT_RES4: 0 = NOT_SET 1 = SET
28:23	0x0	EVENT_RES: 0 = NOT_SET 1 = SET
22:19	0x0	EVENT_1: 0 = NOT_SET 1 = SET
18	0x0	EVENT_RES3: power interrupt 0 = NOT_SET 1 = SET
17	0x0	RESERVED: This bit is reserved
16	0x0	RTC: RTC wake 0 = NOT_SET 1 = SET
15:0	0x0	EVENT: pin 0-15 wake 0 = NOT_SET 1 = SET

12.6.7 APBDEV_PMC_SW_WAKE_STATUS_0

This register stores status of the wake events. Latching of the events in software wake status is enabled by the PMC_CNTRL register bit LATCHWAKE_EN.

Latching will stop at a 1-to-0 transition on this bit. An event will be set if the level matches. Masking does not affect this register. A write will reset the set wake events.

PMC Software Wake Status

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RESET_N: External reset 0 = NOT_SET 1 = SET
30	0x0	EVENT_RES1: 0 = NOT_SET 1 = SET
29	0x0	EVENT_RES4: 0 = NOT_SET 1 = SET
28:23	0x0	EVENT_RES: 0 = NOT_SET 1 = SET
22:19	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
18	0x0	EVENT_RES3: Power interrupt 0 = NOT_SET 1 = SET
17	0x0	RESERVED: This bit is reserved
16	0x0	RTC: RTC wake 0 = DISABLE 1 = ENABLE
15:0	0x0	EVENT: Pin 0-15 wake 0 = DISABLE 1 = ENABLE

12.6.8 APBDEV_PMC_DPD_PADS_ORIDE_0

This register enables overriding values from pinmux with values driven by the PMC (sys_clk_req, blink).

If a bit is set to 1, the associated I/O will drive the direct value from the PMC, not the pinmux value. During Deep Sleep mode, the pads with the bit set to 1 will not be in Deep Power Down mode. It will not drive data from mini pad macros stored during sample cycle, but direct data from the PMC.

> ***Note:*** *This register has to be set before entering Deep Sleep mode.*

DPD Pads Override

Offset: 0x1c | Read/Write: R/W | Reset: 0x00200000 (0b00000000001000000000000000000000)

Bit	Reset	Description
31:22	0x0	RESERVED_2: This bit is reserved

Bit	Reset	Description
21	0x1	SYS_CLK: DEFUNCT: override DPD idle state of sys_clk_req pad. sys_clk_req pad is neither pinmuxed nor used as GPIO. DPD_IO[1:0], SEL_DPD, E_DPD controls are tied to "0" for this pad. A and EN for this pad goes directly from PMC to pads. DPD_PADS_ORIDE bit is not needed for this. 0 = DISABLE 1 = ENABLE
20	0x0	BLINK: Override DPD idle state with blink output 0 = DISABLE 1 = ENABLE
19:0	0x0	RESERVED_1: This bit is reserved

12.6.9 APBDEV_PMC_DPD_SAMPLE_0

Setting this register will trigger sampling pads data and direction in which the pad will be driven during Deep Sleep mode.

Before writing to this register, all interfaces going to pads must be set to the ideal "idle" mode, which is expected to be driven by pads when the Tegra X1 chip enters Deep Sleep.

DPS Power Down Sample has to precede Deep Power Down Enable write. The DPD sample should not be deasserted until the transition from Deep Sleep to WB0.

This affects only for MPIO pads.

Note: *Only one sample signal exists to perform sampling while entering DPD mode. Thus all interfaces are sampled at the same time. The sequence of multiple DPD groups requires keeping all interfaces (already in DPD mode) in idle.*

Deep Power Down Sample

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ON: Sets the sampling of pads value 0 = DISABLE 1 = ENABLE

12.6.10 APBDEV_PMC_DPD_ENABLE_0

Setting this register will trigger entering Deep Sleep state. It must be preceded by a DPD_SAMPLE write.

Will cause request for shutting down the system clock and the core req power. Puts the PLLs and I/Os in Deep Power Down mode. Will cut off (clamp) all signals going from core power to AO and pads.

If none of the wake-up events is set, only power-on reset can re-enable access to the chip.

After servicing of the wake-up event is completed, the register should be set back to 0 to complete a Deep Sleep cycle.

Deep Power Down Enable

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	TSC_MULT_EN: TSC multiplier enable. 0: DISABLE (i.e., counter runs at oscillator clock and increments by 1 every clock cycle). 1: ENABLE (i.e., counter runs at 32 kHz and increments by TSC_MULT_VALUE every clock cycle).0 = DISABLE 1 = ENABLE
0	0x0	ON: Sets sampling of pads value 0 = DISABLE 1 = ENABLE

12.6.11 APBDEV_PMC_PWRGATE_TIMER_OFF_0

Specifies the number of APB cycles after which the rail line goes off (turns the power to part of power-gated partition). Each rail controls part of the powered partition. This register should be set before the write to Power Gate Toggle. Shared between all power partitions.

Power Gate Timer Off Register

Offset: 0x28 | Read/Write: R/W | Reset: 0xedcba987 (0b11101101110010111010100110000111)

Bit	Reset	Description
31:28	0xe	RAIL7: Timer value for rail 7
27:24	0xd	RAIL6: Timer value for rail 6
23:20	0xc	RAIL5: Timer value for rail 5
19:16	0xb	RAIL4: Timer value for rail 4
15:12	0xa	RAIL3: Timer value for rail 3
11:8	0x9	RAIL2: Timer value for rail 2
7:4	0x8	RAIL1: Timer value for rail 1
3:0	0x7	RAIL0: Timer value for rail 0

12.6.12 APBDEV_PMC_CLAMP_STATUS_0

This register is kept for backward code compatibility; the timer is based on PWRGATE_TIMER_OFF only.

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	VE2:clamp status of VE2 partition 0 = DISABLE 1 = ENABLE
28	X	DFD: clamp status of DFD partition 0 = DISABLE 1 = ENABLE
27	X	AUD: clamp status of AUD partition 0 = DISABLE 1 = ENABLE
26	X	NVJPG: clamp status of NVJPG partition 0 = DISABLE 1 = ENABLE
25	X	NVDEC: clamp status of NVDEC partition 0 = DISABLE 1 = ENABLE
24	X	IRAM: Clamp status of IRAM partition 0 = DISABLE 1 = ENABLE
23	X	VIC: Clamp status of VIC partition 0 = DISABLE 1 = ENABLE
22	X	XUSBC: Clamp status of XUSBC partition 0 = DISABLE 1 = ENABLE
21	X	XUSBB: Clamp status of XUSBB partition 0 = DISABLE 1 = ENABLE
20	X	XUSBA: Clamp status of XUSBA partition 0 = DISABLE 1 = ENABLE
19	X	DISB: Clamp status of DISB partition 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	X	DIS: Clamp status of DIS partition 0 = DISABLE 1 = ENABLE
17	X	SOR: Clamp status of SOR partition 0 = DISABLE 1 = ENABLE
15	X	C0NC: Clamp status of cluster0 nENABLE CPU partition 0 = DISABLE 1 = ENABLE
14	X	CE0: Clamp status of CE0 partition 0 = DISABLE 1 = ENABLE
11	X	CE3: Clamp status of CE3 partition 0 = DISABLE 1 = ENABLE
10	X	CE2: Clamp status of CE2 partition 0 = DISABLE 1 = ENABLE
9	X	CE1: Clamp status of CE1 partition 0 = DISABLE 1 = ENABLE
8	X	SAX: Clamp status of SAX partition 0 = DISABLE 1 = ENABLE
6	X	MPE: Clamp status of MPE partition 0 = DISABLE 1 = ENABLE
3	X	PCX: Clamp status of PCX partition 0 = DISABLE 1 = ENABLE
2	X	VE: Clamp status of VE partition 0 = DISABLE 1 = ENABLE
0	X	CRAIL: Clamp status of CPU Rail 0 = DISABLE 1 = ENABLE

12.6.13 APBDEV_PMC_PWRGATE_TOGGLE_0

Write to this register will turn power on/off to the specified power-gated partition. Only one partition is turned on/off at a time.

PWRGATE_STATUS should be read to determine the state of the partition before writing to this register.

Turning the partition off will cause automatic clamping of all signals generated by the power-gated partition being turned off. Before the partition is turned off, all clocks to the partition should be stopped, and the reset to the partition should be asserted.

Turning the partition on will not remove clamping. Clamping is removed only after a REMOVE_CLAMPING_CMD write.

The user must allow a minimum 20 APB clock cycles between consecutive partition Power-Gate Toggle requests.

The role of the START bit has changed from prior Tegra devices. The START bit is cleared by hardware when the PMC accepts the request to power-gate or unpower-gate the partition. So in order to power-gate/unpower-gate a partition, software needs to do the following:

- Check to see if the partition is already in the correct state, by looking at the PWRGATE_STATUS register.
- If the partition is not in the correct state, software reads the PWRGATE_TOGGLE register to see if the START bit is 0.
- If the START bit is not 0, software polls until the start bit is set to 0.
- Then program the PWRGATE_TOGGLE register with the START bit set to 1 and choose the required partition to be power-gated.

- Ideally, software can poll to check the START bit going back to 0, which indicates that the PMC has accepted the request, and then poll the STATUS register to make sure the required partition is power-gated/unpower-gated.

Power Gate Toggle

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxx00000)

Bit	Reset	Description
8	0x0	START: Start power down/up 0 = DISABLE 1 = ENABLE
4:0	0x0	PARTID: ID of the partition to be toggled 0 = CRAIL 2 = Video Encode 3 = PCX 6 = MPEG Encode 8 = SAX 9 = CE1 10 = CE2 11 = CE3 14 = CE0 15 = C0NC 17 = SOR 18 = DIS 19 = DISB 20 = XUSBA 21 = XUSBB 22 = XUSBC 23 = VIC 24 = IRAM 25 = NVDEC 26 = NVJPG 27 = AUD 28 = DFD 29 = VE2

12.6.14 APBDEV_PMC_REMOVE_CLAMPING_CMD_0

This is a bitmap with one bit per power partition controller by the PMC.

When written to 1b, the PMC removes the clamp signals to the corresponding partition. If the partition is not powered on, the register write will be ignored.

The bit is automatically reset to 0b when the clamping has been removed. Software is responsible for writing to this register at the correct time, that is, when the clocks are started but the blocks in that partition are still held in reset.

Remove Clamping

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000x00000x0x0x0x0)

Bit	Reset	Description
29	0x0	VE2:remove clamping to VE2 0 = DISABLE 1 = ENABLE
28	0x0	DFD: remove clamping to DFD 0 = DISABLE 1 = ENABLE
27	0x0	AUD: remove clamping to AUD 0 = DISABLE 1 = ENABLE
26	0x0	NVJPG: remove clamping to NVJPG 0 = DISABLE 1 = ENABLE
25	0x0	NVDEC: remove clamping to NVDEC 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	0x0	IRAM: Remove clamping from IRAM partition 0 = DISABLE 1 = ENABLE
23	0x0	VIC: Remove clamping from VIC partition 0 = DISABLE 1 = ENABLE
22	0x0	XUSBC: Remove clamping from XUSBC partition 0 = DISABLE 1 = ENABLE
21	0x0	XUSBB: Remove clamping from XUSBB partition 0 = DISABLE 1 = ENABLE
20	0x0	XUSBA: Remove clamping from XUSBA partition 0 = DISABLE 1 = ENABLE
19	0x0	DISB: Remove clamping from DISB partition 0 = DISABLE 1 = ENABLE
18	0x0	DIS: Remove clamping from DIS partition 0 = DISABLE 1 = ENABLE
17	0x0	SOR: Remove clamping from SOR partition 0 = DISABLE 1 = ENABLE
16	0x0	RESERVED_C1NC: Reserved
15	0x0	C0NC: Remove clamping from cluster 0 non-CPU partition 0 = DISABLE 1 = ENABLE
14	0x0	CE0: Remove clamping from CE0 partition 0 = DISABLE 1 = ENABLE
12	0x0	RESERVED_CELP: Reserved
11	0x0	CE3: Remove clamping from CE3 partition 0 = DISABLE 1 = ENABLE
10	0x0	CE2: Remove clamping from CE2 partition 0 = DISABLE 1 = ENABLE
9	0x0	CE1: Remove clamping from CE1 partition 0 = DISABLE 1 = ENABLE
8	0x0	SAX: Remove clamping from SAX partition 0 = DISABLE 1 = ENABLE
6	0x0	MPE: Remove clamping from MPE_CACHE partition 0 = DISABLE 1 = ENABLE
4	0x0	PCX: Remove clamping from PCX partition 0 = DISABLE 1 = ENABLE
2	0x0	VE: Remove clamping from VE partition 0 = DISABLE 1 = ENABLE
0	0x0	CRAIL: Remove clamping from CPU Rail domain 0 = DISABLE 1 = ENABLE

12.6.15 APBDEV_PMC_PWRGATE_STATUS_0

This read-only register displays the status of the power partitions. This register should be read before writing to PWRGATE_TOGGLE.

Power Gate Status

Offset: 0x38 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	VE2: Status of VE2 partition 0 = OFF 1 = ON
28	X	DFD: Status of DFD partition 0 = OFF 1 = ON
27	X	AUD: Status of AUD partition 0 = OFF 1 = ON
26	X	NVJPG: Status of NVJPG partition 0 = OFF 1 = ON
25	X	NVDEC: Status of NVDEC partition 0 = OFF 1 = ON
24	X	IRAM: Status of IRAM partition 0 = OFF 1 = ON
23	X	VIC: Status of VIC partition 0 = OFF 1 = ON
22	X	XUSBC: Status of XUSBC partition 0 = OFF 1 = ON
21	X	XUSBB: Status of XUSBB partition 0 = OFF 1 = ON
20	X	XUSBA: Status of XUSBA partition 0 = OFF 1 = ON
19	X	DISB: Status of DISB partition 0 = OFF 1 = ON
18	X	DIS: Status of DIS partition 0 = OFF 1 = ON
17	X	SOR: Status of SOR partition 0 = OFF 1 = ON
15	X	C0NC: Status of cluster0 non-CPU partition 0 = OFF 1 = ON
14	X	CE0: Status of CE0 partition 0 = OFF 1 = ON
11	X	CE3: Status of CE3 partition 0 = OFF 1 = ON
10	X	CE2: Status of CE2 partition 0 = OFF 1 = ON

Bit	Reset	Description
9	X	CE1: Status of CE1 partition 0 = OFF 1 = ON
8	X	SAX: Status of SAX partition 0 = OFF 1 = ON
6	X	MPE: Status of MPE partition 0 = OFF 1 = ON
3	X	PCX: Status of PCX partition 0 = OFF 1 = ON
2	X	VE: Status of VE partition 0 = OFF 1 = ON
0	X	CRAIL: Status of CPU Rail 0 = OFF 1 = ON

12.6.16 APBDEV_PMC_PWRGOOD_TIMER_0

This register programs the length of the wake-up reset, asserted after wake-up from Deep Sleep. This register should be set before entering Deep Sleep mode.

Programmed values depend on the properties of the PMIC.

Timer value x 30.51 µs = reset pulse length

Power Good Timer

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x003f007f (0bxxxxxxxx00111111000000001111111)

Bit	Reset	Description
23:16	0x3f	OSC_PREPWR: OSC clock stabilization timer prior to SoC rail pwr-req assertion. The timer value is 4*OSC_PREPWR+1 number of 32.768 kHz clock cycles; that is, the timer value is (OSC_PREPWR*122.07)+30.518 µs.
15:8	0x0	OSC_POSTPWR: OSC clock stabilization timer after SoC rail power is stabilized. The timer value is 4*OSC_POSTPWR+1 number of 32.768 kHz clock cycles; that is, the timer value is (OSC_POSTPWR*122.07)+30.518 µs.
7:0	0x7f	PWRGOOD: SoC rail power-on stabilization timer. The timer value is PWRGOOD+1 number of 32.768 kHz clock cycles; that is, the timer value is (PWRGOOD*30.52)+30.518 µs.

12.6.17 APBDEV_PMC_BLINK_TIMER_0

Will output value to pad only if the PMC Control register has BLINK_EN set and DPD_PADS_ORIDE has blink bit set.

Setting bit 15 to 0 will output 32 kHz clock (the registers above still have to be set)

- (On time DATA_ON * 16) x 30.51 µs
- (Off time DATA_OFF * 16) x 30.51 µs.

Blinker Timer for External Blinker

Offset: 0x40 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	DATA_OFF: Time off
15	0x1	FORCE_BLINK: set to 0 for 32 KHz clock
14:0	0x7fff	DATA_ON: Time on

12.6.18 APBDEV_PMC_NO_IOPOWER_0

The APBDEV_PMC_NO_IOPOWER register contains control bits that are used to electrically isolate in-bound paths (for example, signals entering the core voltage domain from I/O power domain) in pads, when their I/O power supply (or VDDP) is powered OFF. Before an I/O rail is powered off, this bit must be set to HIGH. Similarly, after the I/O rail is powered UP, this bit needs to be cleared to the LOW state so that the pad can be used for functional purposes.

No I/O Power Register

Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00010080 (0bxxxxxx00000000x1x000000x1x0xx0x0)

Bit	Reset	Description
25	0x0	DP: rail DP IOs – Boot ROM will clear this before using eMMC 0 = DISABLE 1 = ENABLE
24	0x0	SDMMC2: rail SDMMC2 I/Os 0 = DISABLE 1 = ENABLE
23	0x0	SPI_HV: rail AO I/Os 0 = DISABLE 1 = ENABLE
22	0x0	SPI: rail AO I/Os 0 = DISABLE 1 = ENABLE
21	0x0	GPIO: rail AO I/Os 0 = DISABLE 1 = ENABLE
20	0x0	DMIC: rail AO I/Os 0 = DISABLE 1 = ENABLE
19	0x0	DBG: rail AO I/Os 0 = DISABLE 1 = ENABLE
18	0x0	AUDIO_HV: rail AO I/Os 0 = DISABLE 1 = ENABLE
16	0x1	MEM_COMP:MEM0 ADDR1 (comp cell I/Os) 0 = DISABLE 1 = ENABLE
14	0x0	SDMMC4: rail SDMMC4 I/Os 0 = DISABLE 1 = ENABLE
13	0x0	SDMMC3: rail SDMMC3 I/Os 0 = DISABLE 1 = ENABLE
12	0x0	SDMMC1: rail SDMMC1 I/Os 0 = DISABLE 1 = ENABLE
11	0x0	PEX_CNTRL: pex_cpex_cntrl rail I/Os 0 = DISABLE 1 = ENABLE
10	0x0	CAM: Cam rail I/Os 0 = DISABLE 1 = ENABLE
9	0x0	MIPI: MIPI rail I/Os 0 = DISABLE 1 = ENABLE
7	0x1	MEM: rail memory I/Os 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	AUDIO: rail I2S 0 = DISABLE 1 = ENABLE
2	0x0	UART: rail DBG I/Os for UART 0 = DISABLE 1 = ENABLE
0	0x0	SYS: rail AO I/Os. SYS rail is a pre-boot rail that works without any programming intervention 0 = DISABLE 1 = ENABLE

12.6.19 APBDEV_PMC_PWR_DET_0

Active high, sets power detection for nine power rails only. - MIPI does not have power detect.

The proper sequence for turning on power detects:

- Write 1 to PWR_DET register for fields requiring power detect
- Allow for 3 µs delay (in APB clock cycles)
- Write 1 to PWR_DET_LATCH

For turning PWR_DET off:

- Write 0 to PWR_DET_LATCH
- No delay is necessary
- Write 0 to PWR_DET

Power Detect

Offset: 0x48 │ Read/Write: R/W │ Reset: 0x00feb425 (0bxxxxxxxx1111111x1x1101xxxx1x0101)

Bit	Reset	Description
23	0x1	SPI_HV: rail AO I/Os 0 = DISABLE 1 = ENABLE
22	0x1	SPI: rail AO I/Os 0 = DISABLE 1 = ENABLE
21	0x1	GPIO: rail AO I/Os 0 = DISABLE 1 = ENABLE
20	0x1	DMIC: rail AO I/Os 0 = DISABLE 1 = ENABLE
19	0x1	DBG: rail AO I/Os 0 = DISABLE 1 = ENABLE
18	0x1	AUDIO_HV: rail AO I/Os 0 = DISABLE 1 = ENABLE
17	0x1	RESERVED_4:RESERVED - This bit is reserved
15	0x1	HV: DEFUNCT
13	0x1	SDMMC3: rail SDMMC3 I/Os 0 = DISABLE 1 = ENABLE
12	0x1	SDMMC1: rail SDMMC1 I/Os 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	PEX_CNTRL: rail PEX_CNTRL 0 = DISABLE 1 = ENABLE
10	0x1	CAM: rail Cam 0 = DISABLE 1 = ENABLE
5	0x1	AUDIO: rail I2S I/Os 0 = DISABLE 1 = ENABLE
3	0x0	BB: This bit is reserved
2	0x1	UART: rail DBG I/Os 0 = DISABLE 1 = ENABLE
1	0x1	RESERVED_0: RESERVED - This bit is reserved
0	0x1	SYS: rail AO I/Os 0 = DISABLE 1 = ENABLE

12.6.20 APBDEV_PMC_PWR_DET_LATCH_0

Latches power detect for power rails enabled by Power Detect register.

Power Detect Latch

Offset: 0x4c │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	LATCH: power detect latch, latches value as long set to 1 0 = DISABLE 1 = ENABLE

12.6.21 APBDEV_PMC_SCRATCH0_0

Scratch Register

Scratch registers for restoring context after wake-up. On a cold power up, the content of register 0 will be reset to 0x0.

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH0: General-purpose register storage

12.6.22 APBDEV_PMC_SCRATCH1_0

Scratch Register

Offset: 0x54 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH1: General-purpose register storage

12.6.23 APBDEV_PMC_SCRATCH2_0

Scratch Register

Offset: 0x58 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH2: General-purpose register storage

12.6.24 APBDEV_PMC_SCRATCH3_0

Scratch Register

Offset: 0x5c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH3: General-purpose register storage

12.6.25 APBDEV_PMC_SCRATCH4_0

Scratch Register

Offset: 0x60 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH4: General-purpose register storage

12.6.26 APBDEV_PMC_SCRATCH5_0

Scratch Register

Offset: 0x64 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH5: General-purpose register storage

12.6.27 APBDEV_PMC_SCRATCH6_0

Scratch Register

Offset: 0x68 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH6: General-purpose register storage

12.6.28 APBDEV_PMC_SCRATCH7_0

Scratch Register

Offset: 0x6c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH7: General-purpose register storage

12.6.29 APBDEV_PMC_SCRATCH8_0

Scratch Register

Offset: 0x70 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH8: General-purpose register storage

12.6.30 APBDEV_PMC_SCRATCH9_0

Scratch Register

Offset: 0x74 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH9: General-purpose register storage

12.6.31 APBDEV_PMC_SCRATCH10_0

Scratch Register

Offset: 0x78 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH10: General-purpose register storage

12.6.32 APBDEV_PMC_SCRATCH11_0

Scratch Register

Offset: 0x7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH11: General-purpose register storage

12.6.33 APBDEV_PMC_SCRATCH12_0

Scratch Register

Offset: 0x80 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH12: General-purpose register storage

12.6.34 APBDEV_PMC_SCRATCH13_0

Scratch Register

Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH13: General-purpose register storage

12.6.35 APBDEV_PMC_SCRATCH14_0

Scratch Register

Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH14: General-purpose register storage

12.6.36 APBDEV_PMC_SCRATCH15_0

Scratch Register

Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH15: General-purpose register storage

12.6.37 APBDEV_PMC_SCRATCH16_0

Scratch Register

Offset: 0x90 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH16: General-purpose register storage

12.6.38 APBDEV_PMC_SCRATCH17_0

Scratch Register

Offset: 0x94 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH17: General-purpose register storage

12.6.39 APBDEV_PMC_SCRATCH18_0

Scratch Register

Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH18: General-purpose register storage

12.6.40 APBDEV_PMC_SCRATCH19_0

Scratch Register

Offset: 0x9c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH19: General-purpose register storage

12.6.41 APBDEV_PMC_SCRATCH20_0

Scratch Register

Offset: 0xa0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH20: General-purpose register storage

12.6.42 APBDEV_PMC_SCRATCH21_0

Scratch Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH21: General-purpose register storage

12.6.43 APBDEV_PMC_SCRATCH22_0

Scratch Register

Offset: 0xa8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH22: General-purpose register storage

12.6.44 APBDEV_PMC_SCRATCH23_0

Scratch Register

Offset: 0xac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH23: General-purpose register storage

12.6.45 APBDEV_PMC_SECURE_SCRATCH0_0

Secure Scratch Register

Offset: 0xb0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH0

12.6.46 APBDEV_PMC_SECURE_SCRATCH1_0

Secure Scratch Register

Offset: 0xb4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH1

12.6.47 APBDEV_PMC_SECURE_SCRATCH2_0

Secure Scratch Register

Offset: 0xb8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH2

12.6.48 APBDEV_PMC_SECURE_SCRATCH3_0

Secure Scratch Register

Offset: 0xbc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH3

12.6.49 APBDEV_PMC_SECURE_SCRATCH4_0

Secure Scratch Register

Offset: 0xc0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH4

12.6.50 APBDEV_PMC_SECURE_SCRATCH5_0

Secure Scratch Register

Offset: 0xc4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH5

12.6.51 APBDEV_PMC_CPUPWRGOOD_TIMER_0

This register is reserved.

Offset: 0xc8 | Read/Write: R/W | Reset: 0x0000ffff (0b00000000000000001111111111111111)

Bit	Reset	Description
31:0	0xffff	Reserved.

12.6.52 APBDEV_PMC_CPUPWROFF_TIMER_0

This register is reserved.

Offset: 0xcc | Read/Write: R/W | Reset: 0x0000ffff (0b00000000000000001111111111111111)

Bit	Reset	Description
31:0	0xffff	Reserved.

12.6.53 APBDEV_PMC_PG_MASK_0

Offset: 0xd0 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	PCX: Mask PCX rail
23:16	0xff	RESERVED_VD: Reserved
15:8	0xff	VE: Mask VE rail
7:0	0xff	RESERVED_TD: Reserved

12.6.54 APBDEV_PMC_PG_MASK_1_0

Offset: 0xd4 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	SAX: Mask SAX rail
23:16	0xff	RESERVED_HEG: Reserved
15:8	0xff	MPE: Mask MPE rail
0	0xff	RESERVED_C0L2: Reserved

12.6.55 APBDEV_PMC_AUTO_WAKE_LVL_0

Note: *This register is not needed.*

The wake levels are snapped (during deep sleep, or LP0, entry) just before entering DPD by default.

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SMPL: Causes PMC to sample the wake pads 0 = DISABLE 1 = ENABLE

12.6.56 APBDEV_PMC_AUTO_WAKE_LVL_MASK_0

This register is used by software to enable sampling of the wake pads.

Setting a '1' enables auto wake level sampling for the associated pad.

Offset: 0xdc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

12.6.57 APBDEV_PMC_WAKE_DELAY_0

WAKE_DELAY should be a non-zero value.

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	VALUE

12.6.58 APBDEV_PMC_PWR_DET_VAL_0

This register is used to override the power-detect cells and manually set the values. A write to this register also causes the values from the power-detect cells to be captured and which can be subsequently read out.

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00fcbc2d (0bxxxxxxxx1111110x1x1111xxxx1x1101)

Note: *BCT via bootROM sets the PMR_DET_VAL register at boot time.*

Bit	Reset	Description
23	0x1	SPI_HV: rail AO I/Os 0 = DISABLE 1 = ENABLE
22	0x1	DEFUNCT

Bit	Reset	Description
21	0x1	GPIO: rail AO I/Os 0 = DISABLE 1 = ENABLE
20	0x1	DEFUNCT
19	0x1	DEFUNCT
18	0x1	AUDIO_HV: rail AO I/Os 0 = DISABLE 1 = ENABLE
17	0x0	DEFUNCT
15	0x1	DEFUNCT
13	0x1	SDMMC3: rail SDMMC3 I/Os 1 = ENABLE 0 = DISABLE
12	0x1	SDMMC1: rail SDMMC1 I/Os 1 = ENABLE 0 = DISABLE
11	0x1	DEFUNCT
10	0x1	DEFUNCT
5	0x1	DEFUNCT
3	0x1	DEFUNCT
2	0x1	DEFUNCT
1	0x1	DEFUNCT
0	0x1	DEFUNCT

12.6.59 APBDEV_PMC_DDR_PWR_0

This register is used to program the "E_18V" pin of the DDR pads.

Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	0x1	SPI:SPI pins 0 = E_12V 1 = E_18V
2	0x1	EMMC2:SDMMC2 pins 0 = E_12V 1 = E_18V
1	0x1	EMMC: GMI pins 0 = E_12V 1 = E_18V
0	0x1	VAL: DVI pins 0 = E_12V 1 = E_18V

12.6.60 APBDEV_PMC_USB_DEBOUNCE_DEL_0

Determines number of 32 kHz clock cycles to debounce USB signal events.

Note: *The programmed debounce values must be greater than 1.*

Reset values are for Cold Hardware Reset only.

Offset: 0xec │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:20	0x0	UHSIC_LINE_DEB_CNT: Number of 32 kHz debounce cycles for UHSIC port 0
19:16	0x0	UTMIP_LINE_DEB_CNT: Number of 32 kHz debounce cycles for UTMIP port 0

Bit	Reset	Description
15:0	0x0	VAL: Debounce period for ID and VBUS events on all USB ports.

12.6.61 APBDEV_PMC_USB_AO_0

Power downs for various USB features controlled by the PMC.

Each UTMIP has the following power downs for features that can lead to wake-up events.

- USBOP_VAL_PD: Power down for D+ value detector (I/O pad)
- USBON_VAL_PD: Power down for D- value detector (I/O pad)
- ID_PD: Power down for ID detector (bias pad)
- VBUS_WAKEUP_PD: Power down for VBUS detector (bias pad)

Each UHSIC port has the following power downs for features that can lead to wake-up events:

- STROBE_VAL_PD: Power down for STROBE VALUE detector
- DATA_VAL_PD: Power down for DATA VALUE detector

Note: *These HSIC pins have not been added to the pad yet.*

Line Debounce Periods for UTMIP and HSIC PORTS

Offset: 0xf0 | Read/Write: R/W | Reset: 0x01ffffff (0bxxxxxxx11111111111111111111111)

Bit	Reset	Description
24	0x1	DATA1_VAL_PD_P0: Power Down D- DATA1_VAL receiver for UHSIC P0
23	0x1	ID_PD_P3: Power Down ID Wake up for UTMIP P3
22	0x1	VBUS_WAKEUP_PD_P3: Power Down Vbus Wake Up for UTMIP P3
21	0x1	USBON_VAL_PD_P3: Power Down D- USBOP_VAL receiver for UTMIP P3
20	0x1	USBOP_VAL_PD_P3: Power Down D+ USBOP_VAL receiver for UTMIP P3
19:18	0x3	UHSIC_RESERVED_P1: 2 bits reserved for UHSIC P1
17	0x1	DATA_VAL_PD_P1: Power Down D- DATA0_VAL receiver for UHSIC P1
16	0x1	STROBE_VAL_PD_P1: Power Down D+ STROBE_VAL receiver for UHSIC P1
15:14	0x3	UHSIC_RESERVED_P0: 2 bits reserved for UHSIC P1
13	0x1	DATA_VAL_PD_P0: Power Down D- DATA0_VAL receiver for UHSIC P0
12	0x1	STROBE_VAL_PD_P0: Power Down D+ STROBE_VAL receiver for UHSIC P0
11	0x1	ID_PD_P2: Power Down ID Wake up for UTMIP
10	0x1	VBUS_WAKEUP_PD_P2: Power Down Vbus Wake Up for UTMIP P2
9	0x1	USBON_VAL_PD_P2: Power Down D- USBOP_VAL receiver for UTMIP P0
8	0x1	USBOP_VAL_PD_P2: Power Down D+ USBOP_VAL receiver for UTMIP P0
7	0x1	ID_PD_P1: Power Down ID Wake up for UTMIP P1
6	0x1	VBUS_WAKEUP_PD_P1: Power Down Vbus Wake Up for UTMIP P1
5	0x1	USBON_VAL_PD_P1: Power Down D- USBOP_VAL receiver for UTMIP P0
4	0x1	USBOP_VAL_PD_P1: Power Down D+ USBOP_VAL receiver for UTMIP P0
3	0x1	ID_PD_P0: Power Down ID Wake up for UTMIP
2	0x1	VBUS_WAKEUP_PD_P0: Power Down Vbus Wake Up for UTMIP P0
1	0x1	USBON_VAL_PD_P0: Power Down D- USBOP_VAL receiver for UTMIP P0
0	0x1	USBOP_VAL_PD_P0: Power Down D+ USBOP_VAL receiver for UTMIP P0

12.6.62 APBDEV_PMC_CRYPTO_OP_0

A complete solution requires a "semi-sticky" bit in the always-on domain. The Boot ROM would clear this semi-sticky bit for cold boots, but use its value to propagate the crypto-disable flag across deep sleep. (The Boot ROM always requires crypto functionality, so a pure hardware solution probably isn't reasonable.)

The Boot ROM would be able to clear (to zero) and read the sticky bit, but outside the Boot ROM users would only be able to set (to one) and read the sticky bit. On cold boot, the Boot ROM would always clear the stick bit. On WB0, the Boot ROM would read the sticky bit and copy it's setting to the crypto disable flag.

Crypto Engine Disable Sticky Bit

Offset: 0xf4 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	VAL: Disabled by default 0 = ENABLE 1 = DISABLE

12.6.63 APBDEV_PMC_PLLP_WB0_OVERRIDE_0

Master control for all WB0 PLL overrides.

Offset: 0xf8 │ Read/Write: R/W │ Reset: 0x00018000 (0bxxxxxxxxxxxxxxx11000000000000000)

Bit	Reset	Description
16	0x1	PLLP_IDDQ: Enabling during LP0 exit, PMC should program IDDQ bit
15	0x1	PLLU_IDDQ: Enabling during LP0 exit, PMC should program IDDQ bit
14	0x0	DUAL_PLLM_IDDQ: Drives IDDQ input to dual PLLM macro through the port pmc2car_pllm_syncmux_ctrl
13	0x0	DUAL_PLLM_SELVCO: Drives SELVCO input to dual PLLM macro through the port pmc2car_pllm_selvco
12	0x0	PLLM_ENABLE: 1 = enable PLLM, 0 = disable PLLM
11	0x0	PLLM_OVERRIDE_ENABLE: 1 = override CAR PLLM setting, 0 = no override. PLLM WB to programmable frequency.
10	0x0	PLLU_ENABLE: 1 = enable PLLU, 0 = disable PLLU.
9	0x0	PLLU_OVERRIDE_ENABLE: 1 = override CAR PLLU setting, 0 = no override. PLLU WB to fixed frequency
8	0x0	OSC_OVERRIDE_ENABLE: 1 = override CAR OSC setting, 0 = no override. Controls OSC_FREQ and PLL_REF_DIV
7:6	0x0	PLL_REF_DIV: PLL reference clock divide for all PLLs. 00 = /1, 01 = /2, 10 = /4, 11 = reserve.
5:2	0x0	OSC_FREQ: Oscillator frequency for shared PLL reference 0000 = 13MHz 0100 = 19.2MHz 1000 = 12MHz 1100 = 26MHz 0001 = 16.8MHz 0101 = 38.4MHz 1001 = 48.0MHz Note: At the time of writing, only 38.4 MHz is supported by NVIDIA for a crystal (or external clock) frequency. Other frequencies mentioned in this section are not supported.
1	0x0	PLLP_ENABLE: 1 = enable PLLP, 0 = disable PLLP
0	0x0	PLLP_OVERRIDE_ENABLE: 1 = override CAR PLLP setting, 0 = no override. PLLP WB to fixed frequency.

12.6.64 APBDEV_PMC_SCRATCH24_0

Scratch Register

Offset: 0xfc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH24: General-purpose register storage

12.6.65 APBDEV_PMC_SCRATCH25_0

Scratch Register

Offset: 0x100 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH25: General-purpose register storage

12.6.66 APBDEV_PMC_SCRATCH26_0

Scratch Register

Offset: 0x104 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH26: General-purpose register storage

12.6.67 APBDEV_PMC_SCRATCH27_0

Scratch Register

Offset: 0x108 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH27: General-purpose register storage

12.6.68 APBDEV_PMC_SCRATCH28_0

Scratch Register

Offset: 0x10c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH28: General-purpose register storage

12.6.69 APBDEV_PMC_SCRATCH29_0

Scratch Register

Offset: 0x110 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH29: General-purpose register storage

12.6.70 APBDEV_PMC_SCRATCH30_0

Scratch Register

Offset: 0x114 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH30: General-purpose register storage

12.6.71 APBDEV_PMC_SCRATCH31_0

Scratch Register

Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH31: General-purpose register storage

12.6.72 APBDEV_PMC_SCRATCH32_0

Scratch Register

Offset: 0x11c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH32: General-purpose register storage

12.6.73 APBDEV_PMC_SCRATCH33_0

Scratch Register

Offset: 0x120 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH33: General-purpose register storage

12.6.74 APBDEV_PMC_SCRATCH34_0

Scratch Register

Offset: 0x124 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH34: General-purpose register storage

12.6.75 APBDEV_PMC_SCRATCH35_0

Scratch Register

Offset: 0x128 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH35: General-purpose register storage

12.6.76 APBDEV_PMC_SCRATCH36_0

Scratch Register

Offset: 0x12c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH36: General-purpose register storage

12.6.77 APBDEV_PMC_SCRATCH37_0

Scratch Register

Offset: 0x130 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH37: General-purpose register storage

12.6.78 APBDEV_PMC_SCRATCH38_0

Scratch Register

Offset: 0x134 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH38: General-purpose register storage

12.6.79 APBDEV_PMC_SCRATCH39_0

Scratch Register

Offset: 0x138 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH39: General-purpose register storage

12.6.80 APBDEV_PMC_SCRATCH40_0

Scratch Register

Offset: 0x13c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH40: General-purpose register storage

12.6.81 APBDEV_PMC_SCRATCH41_0

Scratch Register

Offset: 0x140 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH41: General-purpose register storage

12.6.82 APBDEV_PMC_SCRATCH42_0

Scratch Register

Offset: 0x144 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH42: General-purpose register storage

12.6.83 APBDEV_PMC_BONDOUT_MIRROR0_0

Secure Scratch Register

Offset: 0x148 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR0

12.6.84 APBDEV_PMC_BONDOUT_MIRROR1_0

Secure Scratch Register

Offset: 0x14c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR1

12.6.85 APBDEV_PMC_BONDOUT_MIRROR2_0

Secure Scratch Register

Offset: 0x150 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR2

12.6.86 APBDEV_PMC_SYS_33V_EN_0

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	VAL: 1 = 3.3V 0 = 1.8V

12.6.87 APBDEV_PMC_BONDOUT_MIRROR_ACCESS_0

On separate reset (same as the CAR) disables access (read/write to secure scratch registers)

Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BREAD: disable read from bond out secure registers 0 = OFF 1 = ON
0	0x0	BWRITE: disable write to bond out secure registers 0 = OFF 1 = ON

12.6.88 APBDEV_PMC_GATE_0

This register is used for software controlled synchronization between APB domain and the 32 kHz domain. Software is expected to set the GAKE_WAKE/GATE_DBNS field high to cut the 32 kHz gated clock before updating WAKE_LVL, AUTO_WAKE_LVL, WAKE_MASK and USB_DEBOUNCE_DEL registers. After these registers have been written the GATE_WAKE/GATE_DBNS bit should be written back to '0' to enable the 32 kHz gated clock.

This gates the 32 kHz clock to just the flops that store the above fields.

Offset: 0x15c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	GATE_DBNS: 0 = OFF 1 = ON
0	0x0	GATE_WAKE: 0 = OFF 1 = ON

12.6.89 APBDEV_PMC_WAKE2_MASK_0

Auto-wake is not present for wake2 events in Tegra X1 devices.

The APBDEV_PMC_WAKE2 registers handle wake events whose number exceeds 32. For wake events whose number is less than or equal to 32, refer to the APBDEV_PMC_WAKE registers above.

Offset: 0x160 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EVENT_REST_3:Rest event 3
30:27	0x0	EVENT_REST_2:Rest events 2
26:12	0x0	EVENT_REST_1: 0 = DISABLE 1 = ENABLE
11:7	0x0	EVENT_REST: 0 = DISABLE 1 = ENABLE
6:5	0x0	EVENT_1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
4:0	0x0	EVENT: 0 = DISABLE 1 = ENABLE

12.6.90 APBDEV_PMC_WAKE2_LVL_0

This register sets the active level for a wake event. It causes an exit from the Deep Sleep state if the input signal level matches the level set in this register and if WAKE2_MASK is set for the event to 1. A level is not needed for 4 line wakeup events; the level is always 1.

PMC Wake Level

Offset: 0x164 │ Read/Write: R/W │ Reset: 0xf9ffffe7 (0b11111001111111111111111111100111)

Bit	Reset	Description
31	0x1	EVENT_REST_3
30:27	0xf	EVENT_REST_2
26:12	0x1fff	EVENT_REST_1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

Bit	Reset	Description
11:7	0x1f	EVENT_REST: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
6:5	0x3	EVENT_1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
4:0	0x7	EVENT: Pin 0-2 wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

12.6.91 APBDEV_PMC_WAKE2_STATUS_0

This register stores status of the wake events. An event is set if the level matches and is not masked.

Writes will reset the set wake event.

- WAKE2_STATUS[0] = als_prox_int_IB
- WAKE2_STATUS[1] = temp_alert_IB
- WAKE2_STATUS[2] = gpio_pz0_IB
- WAKE2_STATUS[3] = gpio_pz1_IB
- WAKE2_STATUS[4] = gpio_pz2_IB
- WAKE2_STATUS[7] = utmip_0_wake
- WAKE2_STATUS[8] = utmip_1_wake
- WAKE2_STATUS[9] = utmip_2_wake
- WAKE2_STATUS[10] = utmip_3_wake
- WAKE2_STATUS[11] = uhsic_wake
- WAKE2_STATUS[12] = wake2pmc_xusb_system_wakeup
- WAKE2_STATUS[13] = sdmmc3_dat1_IB
- WAKE2_STATUS[14] = sdmmc4_dat1_IB
- WAKE2_STATUS[15] = cam_i2c_scl_IB
- WAKE2_STATUS[16] = cam_i2c_sda_IB
- WAKE2_STATUS[17] = gpio_pz5_IB
- WAKE2_STATUS[18] = dp_hpd_0_IB
- WAKE2_STATUS[19] = pwr_int_n_IB
- WAKE2_STATUS[20] = bt_wake_ap_IB
- WAKE2_STATUS[21] = hdmi_int_dp_hpd
- WAKE2_STATUS[22] = usb_vbus_en_0
- WAKE2_STATUS[23] = usb_vbus_en_1
- WAKE2_STATUS[24] = lcd_rst_IB
- WAKE2_STATUS[25] = lcd_gpio1_IB
- WAKE2_STATUS[26] = lcd_gpio2_IB
- WAKE2_STATUS[27] = uart4_cts_IB
- WAKE2_STATUS[28] = batt_bcl_IB (Reserved)

- WAKE2_STATUS[29] = modem_wake_ap_IB
- WAKE2_STATUS[30] = touch_int_IB
- WAKE2_STATUS[31] = motion_int_IB

PMC Wake Status

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EVENT_REST_3
30:27	0x0	EVENT_REST_2
26:12	0x0	EVENT_REST_1: 0 = NOT_SET 1 = SET
11:7	0x0	EVENT_REST: Internally set USB events 0 = NOT_SET 1 = SET
6:5	0x0	EVENT_1: 0 = NOT_SET 1 = SET
4:0	0x0	EVENT: Pin 0-2 wake enable 0 = NOT_SET 1 = SET

12.6.92 APBDEV_PMC_SW_WAKE2_STATUS_0

This register stores the status of the wake events. Latching of the events in software wake status is enabled by the PMC_CNTRL register bit LATCHWAKE_EN. Latching will stop at a 1-to-0 transition on this bit.

An event is set if the level matches. Masking does not affect this register. Writes will reset the set wake events.

PMC Software Wake Status

Offset: 0x16c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EVENT_REST_3
30:27	0x0	EVENT_REST_2
26:12	0x0	EVENT_REST_1: 0 = NOT_SET 1 = SET
11:7	0x0	EVENT_REST
6:5	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
4:0	0x0	EVENT: pin 0-2 wake 0 = DISABLE 1 = ENABLE

12.6.93 APBDEV_PMC_AUTO_WAKE2_LVL_MASK_0

This register is used by s/w to enable sampling of the wake pads. Setting a '1' enables auto wake level sampling for the associated pad.

Offset: 0x170 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EVENT_REST_3
30:27	0x0	EVENT_REST_2
26:12	0x0	EVENT_REST_1

Bit	Reset	Description
11:7	0x0	EVENT_REST
6:0	0x0	VALUE

12.6.94 APBDEV_PMC_PG_MASK_2_0

Power-Gate mask registers for power-gated fields. Need 32 bits for CPU.

Offset: 0x174 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	IRAM: Mask IRAM rail
23:16	0xff	VIC: Mask VIC rail
15:8	0xff	RESERVED_CELP: Mask CELP rail
7:0	0xff	TD2: Mask TD2 rail - Defunct

12.6.95 APBDEV_PMC_PG_MASK_CE1_0

Offset: 0x178 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

12.6.96 APBDEV_PMC_PG_MASK_CE2_0

Offset: 0x17c | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

12.6.97 APBDEV_PMC_PG_MASK_CE3_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

12.6.98 APBDEV_PMC_PWRGATE_TIMER_CE_0_0

Separate power_gate_off, on, for CE counters - 4 bits each.

Offset: 0x184 | Read/Write: R/W | Reset: 0xedcba987 (0b11101101110010111010100110000111)

Bit	Reset	Description
31:28	0xe	RAIL7: Timer Value for rail 7
27:24	0xd	RAIL6: Timer Value for rail 6
23:20	0xc	RAIL5: Timer Value for rail 5
19:16	0xb	RAIL4: Timer Value for rail 4
15:12	0xa	RAIL3: Timer Value for rail 3
11:8	0x9	RAIL2: Timer Value for rail 2
7:4	0x8	RAIL1: Timer Value for rail 1
3:0	0x7	RAIL0: Timer Value for rail 0

12.6.99 APBDEV_PMC_PWRGATE_TIMER_CE_1_0

PWRGATE_TIMER CE_1 removed but kept for backward compatibility.

Offset: 0x188 | Read/Write: R/W | Reset: 0x1240e30a (0bxx0100100100000011100011000010 10)

Bit	Reset	Description
29:24	0x12	RAIL9: Timer Value for rail 9
23:18	0x10	RAIL8: Timer Value for rail 8
17:12	0xe	RAIL7: Timer Value for rail 7
11:6	0xc	RAIL6: Timer Value for rail 6
5:0	0xa	RAIL5: Timer Value for rail 5

12.6.100 APBDEV_PMC_PWRGATE_TIMER_CE_2_0

PWRGATE_TIMER CE_2 removed but kept for backward compatibility.

Offset: 0x18c | Read/Write: R/W | Reset: 0x1c698594 (0bxx0111000110100110000101100101 00)

Bit	Reset	Description
29:24	0x1c	RAIL14: Timer Value for rail 14
23:18	0x1a	RAIL13: Timer Value for rail 13
17:12	0x18	RAIL12: Timer Value for rail 12
11:6	0x16	RAIL11: Timer Value for rail 11
5:0	0x14	RAIL10: Timer Value for rail 10

12.6.101 APBDEV_PMC_PWRGATE_TIMER_CE_3_0

PWRGATE_TIMER CE_3 removed but kept for backward compatibility.

Offset: 0x190 | Read/Write: R/W | Reset: 0x2692281e (0bxx1001101001001000101000000111 10)

Bit	Reset	Description
29:24	0x26	RAIL19: Timer Value for rail 19
23:18	0x24	RAIL18: Timer Value for rail 18
17:12	0x22	RAIL17: Timer Value for rail 17
11:6	0x20	RAIL16: Timer Value for rail 16
5:0	0x1e	RAIL15: Timer Value for rail 15

12.6.102 APBDEV_PMC_PWRGATE_TIMER_CE_4_0

PWRGATE_TIMER CE_4 removed but kept for backward compatibility.

Offset: 0x194 | Read/Write: R/W | Reset: 0x30bacaa8 (0bxx1100001011101011001010101010 00)

Bit	Reset	Description
29:24	0x30	RAIL24: Timer Value for rail 24
23:18	0x2e	RAIL23: Timer Value for rail 23
17:12	0x2c	RAIL22: Timer Value for rail 22
11:6	0x2a	RAIL21: Timer Value for rail 21
5:0	0x28	RAIL20: Timer Value for rail 20

12.6.103 APBDEV_PMC_PWRGATE_TIMER_CE_5_0

PWRGATE_TIMER CE_5 removed but kept for backward compatibility.

Offset: 0x198 | Read/Write: R/W | Reset: 0x3ae36d32 (0bxx1110101110001101101101001100 10)

Bit	Reset	Description
29:24	0x3a	RAIL29: Timer Value for rail 29

Bit	Reset	Description
23:18	0x38	RAIL28: Timer Value for rail 28
17:12	0x36	RAIL27: Timer Value for rail 27
11:6	0x34	RAIL26: Timer Value for rail 26
5:0	0x32	RAIL25: Timer Value for rail 25

12.6.104 APBDEV_PMC_PWRGATE_TIMER_CE_6_0

PWRGATE_TIMER CE_6 removed but kept for backward compatibility.

Offset: 0x19c │ Read/Write: R/W │ Reset: 0x00000f3b (0bxxxxxxxxxxxxxxxxxxxxx111100111011)

Bit	Reset	Description
11:6	0x3c	RAIL31: Timer Value for rail 31
5:0	0x3b	RAIL30: Timer Value for rail 30

12.6.105 APBDEV_PMC_PCX_EDPD_CNTRL_0

Defunct

Offset: 0x1a0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	EN: when set to 1, sets the EDPD to PCX (PLL toggle) 0 = OFF 1 = ON

12.6.106 APBDEV_PMC_OSC_EDPD_OVER_0

This register can be programmed to keep the oscillator ON during LP0 mode. It cuts the latency time on LP0 wake if the oscillator pad does not have to be restarted.

When the oscillator pad is on, during LP0 mode, DSPARE, duty, strength, and bypass are controlled by the fields below.

The oscillator can be in one in four modes during LP0 - in DPD, not in DPD but not enabled, running, or running only when an external request is active.

Offset: 0x1a4 │ Read/Write: R/W │ Reset: 0x0000007e (0bxxxxxxxx00000000000000001111110)

Bit	Reset	Description
23	0x0	CLK_OK: Crystal oscillator clk_ok signal 0 = DISABLE 1 = ENABLE
22	0x0	OSC_CTRL_SELECT: select whether the CAR OSC_CTRL or the PMC OSC_CTRL_OVER affects the oscillator cell. This bit should always be "1" in LP0. 0 = CAR 1 = PMC
21:20	0x0	XO_LP0_MODE: control the behavior of the oscillator during LP0 (assuming OSC_CTRL_SELECT is set to PMC during LP0). Put the oscillator in DPD mode during LP0, bypass DPD, but do not enable the oscillator during LP0, force the oscillator to run during LP0, or run the oscillator when requested by an external clock request pin 0 = DPD. 1 = OFF 2 = ON 3 = EXT_REQ
19:12	0x0	OSCFI_SPARE: Crystal oscillator spare register control.
11:7	0x0	XODS: Crystal oscillator duty cycle control.
6:1	0x3f	XOFS: Crystal oscillator drive strength control.
0	0x0	XOBP: DEFUNCT. Crystal oscillator bypass enable (1 = enable bypass).

12.6.107 APBDEV_PMC_CLK_OUT_CNTRL_0

This register controls 3 clock outputs of the Tegra X1 chip. Each of the clock outputs can be in 1 of 4 modes: not running, running when request is active high, running when request is active low, or always running. The clock output when not running can be tri-stated, high, or low.

The clock source might be from the CAR unit (not available when in LP0 mode), osc, osc_div2, or osc_div4.

The clock source switching on dap_mclk1_out, clk3_out and clk2_out is not glitch free. If dpd_override is not set before entering LP0, the selected clock source will get latched to “0” or “1” in LP0 (which is not glitch free).dpd_override can be “0” only when external device does not need clock in LP0, and it should be in reset before PMC_SAMPLE bit is written by LP0 entry code.

The clock source should only be switched when the pad output is not being used by an external device.

When an external device requests for a clock, the first few clocks should not be used by the external device.

Offset: 0x1a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:22	0x0	CLK3_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV4 3 = CAR (EXTPERIPH1)
21:20	0x0	CLK3_IDLE_STATE: 0 = LOW 1 = HIGH 2 = TRIS
19	0x0	CLK3_RESERVED: reserved
18	0x0	CLK3_FORCE_EN: force the clock to run regardless of ACCEPT_REQ or INVERT_REQ
17	0x0	CLK3_INVERT_REQ:
16	0x0	CLK3_ACCEPT_REQ: 0 = Output clock is driven by CAR (EXTPERIPH1) via the pinmux logic 1 = Output clock is selected by the CLKx_SRC_SEL field bypassing the pinmux logic Note: If the output clock is required in LP0, then the CAR cannot be used and so this bit should be set and the appropriate OSC clock selected via the CLKx_SRC_SEL field.
15:14	0x0	CLK2_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV4 3 = CAR (EXTPERIPH1)
13:12	0x0	CLK2_IDLE_STATE: 0 = LOW 1 = HIGH 2 = TRIS
11	0x0	CLK2_RESERVED: reserved
10	0x0	CLK2_FORCE_EN: force the clock to run regardless of ACCEPT_REQ or INVERT_REQ
9	0x0	CLK2_INVERT_REQ:
8	0x0	CLK2_ACCEPT_REQ: 0 = Output clock is driven by CAR (EXTPERIPH1) via the pinmux logic 1 = Output clock is selected by the CLKx_SRC_SEL field bypassing the pinmux logic Note: If the output clock is required in LP0, then the CAR cannot be used and so this bit should be set and the appropriate OSC clock selected via the CLKx_SRC_SEL field.
7:6	0x0	CLK1_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV4 3 = CAR (EXTPERIPH1)
5:4	0x0	CLK1_IDLE_STATE: state of the output line before clock request is active 0 = LOW 1 = HIGH 2 = TRIS
3	0x0	CLK1_RESERVED: reserved

Bit	Reset	Description
2	0x0	CLK1_FORCE_EN: force the clock to run regardless of ACCEPT_REQ or INVERT_REQ
1	0x0	CLK1_INVERT_REQ:
0	0x0	CLK1_ACCEPT_REQ: 0 = Output clock is driven by CAR (EXTPERIPH1) via the pinmux logic 1 = Output clock is selected by the CLKx_SRC_SEL field bypassing the pinmux logic Note: If the output clock is required in LP0, then the CAR cannot be used and so this bit should be set and the appropriate OSC clock selected via the CLKx_SRC_SEL field.

12.6.108 APBDEV_PMC_SATA_PWRGT_0

This register controls the SATA PLLs

PLLE or PADPLL can be disabled/enabled by a software or hardware request.

The SATA PWRGT is reserved for backward compatibility.

Offset: 0x1ac │ Read/Write: R/W │ Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00111111)

Bit	Reset	Description
7	0x0	SW_PLL_EDPD: Defunct. The SATA PLL is put in the DPD state when this bit is set, independently of power-gating - kept only until drivers are fixed. 0 = OFF 1 = ON
6	0x0	PG_INFO: sm2sata_pg_info
5	0x1	PLLE_IDDQ_OVERRIDE_VALUE: Defunct. 0: The PLLE is powered up. 1: Software can put the PLLE in IDDQ mode by setting this bit and PLLE_IDDQ_SWCTL (default) 0 = OFF 1 = ON
4	0x1	PLLE_IDDQ_SWCTL: 0: Defunct. The PLLE is put in IDDQ mode by hardware (SATA +AFI+CAR) signals. 1: The PLLE is put in IDDQ mode by software -- default 0 = OFF 1 = ON
3	0x1	PADPHY_IDDQ_OVERRIDE_VALUE: Defunct. 0: The PHY is powered up. 1: Software can put the PHY in IDDQ mode by setting this bit and SATA_PADPHY_IDDQ_SWCT (default) 0 = OFF 1 = ON
2	0x1	PADPHY_IDDQ_SWCTL: Defunct. 0 The SATA PHY is put in IDDQ mode by hardware (SATA +AFI+CAR) signals. 1 The SATA PHY is put in IDDQ mode by software (default) 0 = OFF 1 = ON
1	0x1	PADPLL_IDDQ_OVERRIDE_VALUE: Defunct. 0: The pad PLL is powered up 1: Software can put the pad PLL in IDDQ mode by setting this bit and SATA_PADPLL_IDDQ_SWCTL (default) 0 = OFF 1 = ON
0	0x1	PADPLL_IDDQ_SWCTL: Defunct. 0: The SATA pad PLL is put in IDDQ mode by hardware (SATA +AFI+CAR) signals.1: The SATA pad PLL is put in IDDQ mode by software (default) 0 = OFF 1 = ON

12.6.109 APBDEV_PMC_SENSOR_CTRL_0

For sensor shutdown and control.

The sensor control register defines chip behavior when a sensor request is triggered. It can power-gate down all CPUs if enabled. It can trigger a reset (will cause a CPU power request to go down and power-gates down all CPUs)

IMPORTANT-- to preserve RAM content, any reads/writes to scratch registers will be blocked after a sensor triggered reset.

Software must reset BLOCK_SCRATCH_WRITE to enable writes to scratch registers.

Offset: 0x1b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	BLOCK_SCRATCH_WRITE: Reset by user, set by the PMC when entering sensor reset 0 = OFF 1 = ON
1	0x0	ENABLE_RST: Enables reset on sensor going up. 0 = OFF 1 = ON
0	0x0	ENABLE_PG: Power gates CPUs on temperature sensor going up. 0 = OFF 1 = ON

12.6.110 APBDEV_PMC_RST_STATUS_0

Simplified Reset and Reset Source

Offset: 0x1b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	0x0	RST_SOURCE: Source of reset 0 = POR 1 = WATCHDOG 2 = SENSOR 3 = SW_MAIN 4 = LP0 5 = AOTAG

12.6.111 APBDEV_PMC_IO_DPD_REQ_0

Puts I/O rails in or out of DPD mode, even though the chip is not in LP0. Multiple bits are allowed to be set at the same time. No new operation will be triggered until previous one completes. Consecutive operations issued by completion time of first one will be dropped. The register is still updated.

Setting IO_DPD_REQ should be preceded by writing to the DPD_SAMPLE registers (to enable a snapshot of data to be driven) and the SEL_DPD_TIM register. SEL_DPD_TIM is set to a safe value, but it can be lowered if SYS clock is slower (a minimum of 200 ns is required).

Note: *Only one sample signal exists to perform sampling while entering DPD mode. Thus all interfaces are sampled at the same time. The sequence of multiple DPD groups requires keeping all interfaces (already in DPD mode) in idle.*

After writing to IO_DPD_REQ register, software should wait for some time before writing to the IO_DPD2_REQ/IO_DPD3_REQ registers because the same state machine is used for both registers. There is a timing relationship between sel_dpd assertion to e_dpd assertion (similarly sel_dpd deassertion to e_dpd deassertion). This is controlled by a programmable timer (SEL_DPD_TIM register) in PMC. When IO_DPD_REQ register is written, sel_dpd of the corresponding pads get asserted and after the SEL_DPD_TIM delay expiry, e_dpd gets asserted. The state machine goes to idle only after this. So, the time between writes to the IO_DPD_REQ and IO_DPD2_REQ register should be - APB clock frequency multiplied by (SEL_DPD_TIM value plus few APB clocks), for example, APB clock frequency * (SEL_DPD_TIM + 5) The same delay is applicable before software reads the IO_DPD*_STATUS register after the corresponding IO_DPD*_REQ register is written.

DPD Request

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b00x0000xxxxx000000x0000xx0000000)

Bit	Reset	Description
31:30	0x0	CODE: Code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
28	0x0	HDMI: Puts hdmi_pad in/out of Deep Power Down mode 0 = OFF 1 = ON
27	0x0	GPIO: 0 = OFF 1 = ON
26	0x0	DEBUG_NONAO: 0 = OFF 1 = ON
25	0x0	DBG: 0 = OFF 1 = ON
19	0x0	HSIC: puts HSIC rail in/out of Deep Power Down mode 0 = OFF 1 = ON
18	0x0	USB3: puts USB3 pads in/out of Deep Power Down mode 0 = OFF 1 = ON
17	0x0	AUDIO: Puts audio rail in/out of Deep Power Down mode 0 = OFF 1 = ON
16	0x0	RESERVED_VI: Defunct. 0 = OFF 1 = ON
15	0x0	RESERVED_BB: Defunct 0 = OFF 1 = ON
14	0x0	UART: Puts the UART rail in/out of Deep Power Down mode 0 = OFF 1 = ON
12	0x0	USB_BIAS: Puts usb_bias in/out of Deep Power Down mode 0 = OFF 1 = ON
11	0x0	USB2: Puts USB2 in/out of Deep Power Down mode 0 = OFF 1 = ON
10	0x0	USB1:Puts USB1 in/out of Deep Power Down mode 0 = OFF 1 = ON
9	0x0	USB0: Puts USB0 in/out of Deep Power Down mode 0 = OFF 1 = ON
6	0x0	PEX_CLK2: PEX clk2 pad-DPD control 0 = OFF 1 = ON
5	0x0	PEX_CLK1: PEX clk1 pad- DPD control 0 = OFF 1 = ON
4	0x0	PEX_BIAS:PEX bias pad- DPD control 0 = OFF 1 = ON
3	0x0	MIPI_BIAS: Puts mipi_bias in/out of Deep Power Down mode 0 = OFF 1 = ON

Bit	Reset	Description
2	0x0	DSI: (Named DSIA in pinout specification) Puts DSI in/out of Deep Power Down mode 0 = OFF 1 = ON
1	0x0	CSIB: Puts CSIB in/out of Deep Power Down mode 0 = OFF 1 = ON
0	0x0	CSIA: Puts CSIA in/out of Deep Power Down mode 0 = OFF 1 = ON

12.6.112 APBDEV_PMC_IO_DPD_STATUS_0

Reflects the DPD status of each I/O rail.

DPD Status

Offset: 0x1bc │ Read/Write: R/W │ Reset: 0x0000000X (0bxxx0000xxxxx000xx0x0000xx000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	HDMI: hdmi_pad in Deep Power Down mode 0 = OFF 1 = ON
27	RW	0x0	GPIO: 0 = OFF 1 = ON
26	RW	0x0	DEBUG_NONAO: 0 = OFF 1 = ON
25	RW	0x0	DBG: 0 = OFF 1 = ON
19	RW	0x0	HSIC: Puts HSIC rail in Deep Power Down mode 0 = OFF 1 = ON
18	RW	0x0	USB3:Puts USB3 pads in/out of deep power down mode 0 = OFF 1 = ON
17	RW	0x0	AUDIO: Puts audio rail in Deep Power Down mode 0 = OFF 1 = ON
14	RW	0x0	UART: UART rail in Deep Power Down mode 0 = OFF 1 = ON
12	RW	0x0	USB_BIAS: usb_bias in Deep Power Down mode 0 = OFF 1 = ON
11	RW	0x0	USB2: USB2 in Deep Power Down mode 0 = OFF 1 = ON
10	RW	0x0	USB1: USB1 in Deep Power Down mode 0 = OFF 1 = ON
9	RW	0x0	USB0: USB0 in Deep Power Down mode 0 = OFF 1 = ON
6	RW	0x0	PEX_CLK2:PEX clk2 pad-DPD control 0 = OFF 1 = ON
5	RW	0x0	PEX_CLK1:PEX clk1 pad-DPD control 0 = OFF 1 = ON

Bit	R/W	Reset	Description
4	RW	0x0	PEX_BIAS:PEX bias pad-DPD control 0 = OFF 1 = ON
3	RO	X	MIPI_BIAS: mipi_bias in Deep Power Down mode 0 = OFF 1 = ON
2	RO	X	DSI: This is named DSIA in the pinout specification. DSI in Deep Power Down mode 0 = OFF 1 = ON
1	RO	X	CSIB: CSIB in Deep Power Down mode 0 = OFF 1 = ON
0	RO	X	CSIA: CSIA in Deep Power Down mode 0 = OFF 1 = ON

12.6.113 APBDEV_PMC_IO_DPD2_REQ_0

Second set of DPD requests due to additional rails.

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0b000xxx0xxxxx00xx000000000x000000)

Bit	Reset	Description
31:30	0x0	CODE: code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
29	0x0	AUDIO_HV: 0 = OFF 1 = ON
25	0x0	LVDS: for display - eDP 0 = OFF 1 = ON
19	0x0	DP: 0 = OFF 1 = ON
18	0x0	DMIC: 0 = OFF 1 = ON
15	0x0	SPI_HV: 0 = OFF 1 = ON
14	0x0	SPI: 0 = OFF 1 = ON
13	0x0	CSIF: 0 = OFF 1 = ON
12	0x0	CSIE: Puts CSIE in/out of Deep Power Down mode 0 = OFF 1 = ON
11	0x0	CSID: Puts CSID in/out of deep power down mode 0 = OFF 1 = ON
10	0x0	CSIC: Puts CSIC in/out of deep power down mode 0 = OFF 1 = ON
9	0x0	DSID: Puts DSID in/out of Deep Power Down mode 0 = OFF 1 = ON

Bit	Reset	Description
8	0x0	DSIC: Puts DSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
7	0x0	DSIB: Puts DSIB in/out of Deep Power Down mode 0 = OFF 1 = ON
5	0x0	EMMC2: 0 = OFF 1 = ON
4	0x0	CAM: Puts the camera in/out of Deep Power Down mode 0 = OFF 1 = ON
3	0x0	EMMC: Puts SDMMC4 in/out of Deep Power Down mode 0 = OFF 1 = ON
2	0x0	SDMMC3:puts SDMMC3 in/out of Deep Power Down mode 0 = OFF 1 = ON
1	0x0	SDMMC1:puts SDMMC1 in/out of Deep Power Down mode 0 = OFF 1 = ON
0	0x0	PEX_CNTRL: Defunct. 0 = OFF 1 = ON

12.6.114 APBDEV_PMC_IO_DPD2_STATUS_0

DPD2 Status

Offset: 0x1c4 | Read/Write: R/W | Reset: 0x020000X0 (0bxx0xxx1xxxxx00xx00000000xx000000)

Bit	R/W	Reset	Description
29	RW	0x0	AUDIO_HV: 0 = OFF 1 = ON
25	RW	0x1	eDP: for eDP pads 0 = OFF 1 = ON
19	RW	0x0	DP: 0 = OFF 1 = ON
18	RW	0x0	DMIC: 0 = OFF 1 = ON
15	RW	0x0	SPI_HV: 0 = OFF 1 = ON
14	RW	0x0	SPI: 0 = OFF 1 = ON
13	RW	0x0	CSIF: 0 = OFF 1 = ON
12	RW	0x0	CSIE:CSIE in/out of Deep Power Down mode 0 = OFF 1 = ON
11	RW	0x0	CSID:CSID in/out of Deep Power Down mode 0 = OFF 1 = ON
10	RW	0x0	CSIC:CSIC in/out of Deep Power Down mode 0 = OFF 1 = ON

Bit	R/W	Reset	Description
9	RW	0x0	DSID:DSID in/out of Deep Power Down mode 0 = OFF 1 = ON
8	RW	0x0	DSIC:DSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
7	RO	X	DSIB: DSI in Deep Power Down mode 0 = OFF 1 = ON
5	RW	0x0	EMMC2: 0 = OFF 1 = ON
4	RW	0x0	CAM: CAM in/out of Deep Power Down mode 0 = OFF 1 = ON
3	RW	0x0	EMMC: SDMMC4 in/out of Deep Power Down mode 0 = OFF 1 = ON
2	RW	0x0	SDMMC3: SDMMC3 in/out of Deep Power Down mode 0 = OFF 1 = ON
1	RW	0x0	SDMMC1: SDMMC1 in/out of Deep Power Down mode 0 = OFF 1 = ON
0	RW	0x0	PEX_CNTRL: pex_cntrl in/out of Deep Power Down mode 0 = OFF 1 = ON

12.6.115 APBDEV_PMC_SEL_DPD_TIM_0

This timer guarantees proper timing spacing in hardware between the sel_dpd and e_dpd signals issued to pads.

A minimum of 200 ns is required, timer in APB/SYS clock.

Offset: 0x1c8 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	SEL_DPD_TIM: Timer which separates e_dpd deassertion time from sel_dpd deassertion time in apb_clk units.

12.6.116 APBDEV_PMC_VDDP_SEL_0

Power set for new DDR pads. Safe value is 11.

Offset: 0x1cc | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1:0	0x3	DATA: VDDP sel bits to DDR pads

12.6.117 APBDEV_PMC_DDR_CFG_0

Package Type for CAR/PMC Control

Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0x0	BR11_DPD_IO_CMD:DPD_IO_CMD for brick-11 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ

Bit	Reset	Description
29:28	0x0	BR10_DPD_IO_CMD:DPD_IO_CMD for brick-10 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
27:26	0x0	BR9_DPD_IO_CMD:DPD_IO_CMD for brick-9 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
25:24	0x0	BR8_DPD_IO_CMD:DPD_IO_CMD for brick-8 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
23:22	0x0	BR7_DPD_IO_CMD:DPD_IO_CMD for brick-7 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
21:20	0x0	BR6_DPD_IO_CMD:DPD_IO_CMD for brick-6 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
19:18	0x0	BR5_DPD_IO_CMD:DPD_IO_CMD for brick-5 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
17:16	0x0	BR4_DPD_IO_CMD:DPD_IO_CMD for brick-4 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
15:14	0x0	BR3_DPD_IO_CMD:DPD_IO_CMD for brick-3 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
13:12	0x0	BR2_DPD_IO_CMD:DPD_IO_CMD for brick-2 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
11:10	0x0	BR1_DPD_IO_CMD:DPD_IO_CMD for brick-1 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
9:8	0x0	BR0_DPD_IO_CMD:DPD_IO_CMD for brick-0 0 = HOLD_LOW 1 = HOLD_HIGH 2 = HOLD_HIZ
7:4	0x0	RESET_SWIZZLE: DEFUNCT 0 = XM0_RESET 1 = XM0_MBA0 2 = XM0_MBA1 3 = XM0_MBA2 4 = XM0_MA0 5 = XM0_MA1 6 = XM0_MA2 7 = XM0_MA3 8 = XM0_MA10 9 = XM0_MA11 10 = XM0_MA12 11 = XM0_MA13 12 = NONE
3	0x0	DDR_SPARE: Defunct
2	0x0	CHAN_SWIZZLE: Defunct 0 = DISABLE 1 = ENABLE
1	0x0	IF: Defunct 0 = LPDDR2 1 = DDR3

Bit	Reset	Description
0	0x0	RESERVED_PKG

12.6.118 APBDEV_PMC_PLLM_WB0_OVERRIDE_FREQ_0

PLL WB Override Registers to Accelerate Warm Boot Time

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00002a02 (0bxxxxxxxxxxxxxxxx0010101000000010)

Bit	Reset	Description
15:8	0x2a	PLLM_DIVN: PLL feedback divider.
7:0	0x2	PLLM_DIVM: PLL input divider.

12.6.119 APBDEV_PMC_PWRGATE_TIMER_MULT_0

The time for each rail set by PWRGATE_TIMER_OFF will be multiplied by MULT for power-gating up/down any power-gated region except CPU. In the CPU power-gated case, MULT_CPU will be used with PWRGATE_TIMER_CE* registers value. All timers are in sys_clk units.

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x0000001b (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011011)

Bit	Reset	Description
5:3	0x3	MULT_CPU: 0 = ONE 1 = TWO 2 = FOUR 3 = EIGHT 4 = SIXTEEN
2:0	0x3	MULT: 0 = ONE 1 = TWO 2 = FOUR 3 = EIGHT 4 = SIXTEEN

12.6.120 APBDEV_PMC_DSI_SEL_DPD_0

Register to control sel_dpd for the DSI pad. Allows driving LP0 value on the DSI pad beyond LP0 exit. This enables the DSI to be programmed properly at LP0 exit while the LP0 value is still driven by the brick pad.

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	SET_DSID: 0 = OFF 1 = ON
2	0x0	SET_DSIC: 0 = OFF 1 = ON
1	0x0	SET_DSIB: 0 = OFF 1 = ON
0	0x0	SET_DSIA: 0 = OFF 1 = ON

12.6.121 APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0

PMC trigger events for the UTMIP ports as well as the UHSIC port. Sleep walk clearing returns the walk pointer to 00, the first entry in a USB line walk. This is the only way to return a pointer to 0.

Pad configuration captures a set of FLLS parameters prior to resting the port so their value can be retained in deep sleep.

The trigger should only happen on the fields that are set to TRIG (1) when writing. The returned read value should always be all NULL fields.

The FORCE_WALK bits trigger a SLEEP.

Any value read back should be 0.

Triggers for USB Ports

Offset: 0x1ec │ Read/Write: R/W │ Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19	X	UTMIP_CLR_WAKE_ALARM_P3: Clear wake event for UTMIP port 0 0 = NULL 1 = TRIG
18	X	UTMIP_FORCE_WALK_P3: Force pointer walk for UTMIP port 0 0 = NULL 1 = TRIG
17	X	UTMIP_CAP_CFG_P3: Capture pad configuration for UTMIP port 0 0 = NULL 1 = TRIG
16	X	UTMIP_CLR_WALK_PTR_P3: Clear sleep walk pointer for UTMIP port 0 0 = NULL 1 = TRIG
15	X	UHSIC_CLR_WAKE_ALARM_P0: Clear wake event for UHSIC port 0 0 = NULL 1 = TRIG
14	X	UTMIP_CLR_WAKE_ALARM_P2: Clear wake event for UTMIP port 2 0 = NULL 1 = TRIG
13	X	UTMIP_CLR_WAKE_ALARM_P1: Clear wake event for UTMIP port 1 0 = NULL 1 = TRIG
12	X	UTMIP_CLR_WAKE_ALARM_P0: Clear wake event for UTMIP port 0 0 = NULL 1 = TRIG
11	X	UHSIC_FORCE_WALK_P0: Force pointer walk for UHSIC port 0 0 = NULL 1 = TRIG
10	X	UTMIP_FORCE_WALK_P2: Force pointer walk for UTMIP port 2 0 = NULL 1 = TRIG
9	X	UTMIP_FORCE_WALK_P1: Force pointer walk for UTMIP port 1 0 = NULL 1 = TRIG
8	X	UTMIP_FORCE_WALK_P0: Force pointer walk for UTMIP port 0 0 = NULL 1 = TRIG
7	X	UHSIC_RESERVED_P0: Reserved for UHSIC port 0 0 = NULL 1 = TRIG
6	X	UTMIP_CAP_CFG_P2: Capture pad configuration for UTMIP port 2 0 = NULL 1 = TRIG
5	X	UTMIP_CAP_CFG_P1: Capture pad configuration for UTMIP port 1 0 = NULL 1 = TRIG
4	X	UTMIP_CAP_CFG_P0: Capture pad configuration for UTMIP port 0 0 = NULL 1 = TRIG
3	X	UHSIC_CLR_WALK_PTR_P0: Clear sleep walk pointer for UHSIC port 0 0 = NULL 1 = TRIG

Bit	Reset	Description
2	X	UTMIP_CLR_WALK_PTR_P2: Clear sleep walk pointer for UTMIP port 2 0 = NULL 1 = TRIG
1	X	UTMIP_CLR_WALK_PTR_P1: Clear sleep walk pointer for UTMIP port 1 0 = NULL 1 = TRIG
0	X	UTMIP_CLR_WALK_PTR_P0: Clear sleep walk pointer for UTMIP port 0 0 = NULL 1 = TRIG

12.6.122 APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0

Save some critical information about USB port prior to entering DPD in a suspend state.

SPEED:

- HS - Suspend DPD was entered from a high speed mode, restore high speed at DPD exit
- FS - Suspend DPD was entered from a full speed mode, restore full speed at DPD exit
- LS - Suspend DPD was entered from a low speed mode, restore low speed at DPD exit
- RST - Reset Port, Hardware reset or DPD was not entered suspended bus, Reset and Re-enumerate port

SCRATCH -- save other critical information about the port, software choice.

Reset value should read 0x0F0F0F0F, reset should occur on Cold Hardware Reset only

Saved DPD State for all UTMIP and UHSIC Ports

Offset: 0x1f0 | Read/Write: R/W | Reset: 0x0f0f0f0f (0b00001111000011110000111100001111)

Bit	Reset	Description
31	0x0	UHSIC_WAKE_EX_P0: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
30	0x0	UHSIC_IGNORE_MASTER_CFG_P0
29:25	0x7	UHSIC_SCRATCH_P0
24	0x1	UHSIC_MODE_P0: UHSIC Speed prior to DPD 0 = HS 1 = RST
23	0x0	UTMIP_WAKE_EX_P2: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
22	0x0	UTMIP_IGNORE_MASTER_CFG_P2
21:18	0x3	UTMIP_SCRATCH_P2
17:16	0x3	UTMIP_SPEED_P2: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST
15	0x0	UTMIP_WAKE_EX_P1: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
14	0x0	UTMIP_IGNORE_MASTER_CFG_P1
13:10	0x3	UTMIP_SCRATCH_P1
9:8	0x3	UTMIP_SPEED_P1: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST

Bit	Reset	Description
7	0x0	UTMIP_WAKE_EX_P0: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
6	0x0	UTMIP_IGNORE_MASTER_CFG_P0
5:2	0x3	UTMIP_SCRATCH_P0
1:0	0x3	UTMIP_SPEED_P0: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST

12.6.123 APBDEV_PMC_UTMIP_TERM_PAD_CFG_0

Set of termination configuration values for USB I/O pads under DPD mode. These configuration values are programmed, not captured, at a time prior to entering DPD. Capturing them would be complex because it would require costly synchronizers as well as some logic. The range for RCTRL and TCTRL is 0 to 16.

These values need to be converted to a thermal encoding (only one 0 to 1 transition allowed in the word from MSB to LSB). For example:

0 - 0000_0000_0000_0000

1 - 0000_0000_0000_0001

2 - 0000_0000_0000_0011

...

15 - 0111_1111_1111_1111

16 - 1111_1111_1111_1111

I/O Termination Configuration for all UTMIP Ports

Offset: 0x1f8 │ Read/Write: R/W │ Reset: 0x00041041 (0bxxxxxxxxxxxxxx1000001000001000001)

Bit	Reset	Description
18:13	0x20	TCTRL_SW_VAL: Software override for tctrl.
12:7	0x20	TCTRL_VAL: HS termination calibration value
6:1	0x20	PCTRL_VAL: 1.5kOhm pull up calibration value
0	0x1	TERM_SEL: Auto termination enable

12.6.124 APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0

Register that configures the value of the line that could cause a wake-up event.

- ANY -- signifies any line change DM EDGE or DP EDGE for UTMIP.
- NONE -- signifies no wakeup possible.

One can use the two most significant bits (3:2) of the WAKE_VAL to 4x1 select whether:

- 00: Check both bits for value on two LSB (1:0).
- 01: Check bit DM (or strobe for HSIC) against bit 0
- 10: Check bit DP (or DATA for HSIC) against bit 1
- 11: Check for edge on DP against bit 1, edge on DM against bit 0

Sleep Walk Sequence Enables

Offset: 0x1fc | Read/Write: R/W | Reset: 0xc0c0c0c0 (0b11000000110000001100000011000000)

Bit	Reset	Description
31:28	0xc	UHSIC_WAKE_VAL_P0: Line Value Wake Up Condition on UHSIC P0 0 = SD00 1 = SD01 2 = SD10 3 = SD11 4 = S0 5 = S1 8 = D0 10 = D1 12 = NONE 13 = SEDGE 14 = DEDGE 15 = ANY
27:25	0x0	UHSIC_RESERVED_P0: Reserved for later use for UHSIC P0
24	0x0	UHSIC_MASTER_ENABLE_P0: Enable use of master pins on UHSIC P0
23:20	0xc	UTMIP_WAKE_VAL_P2: Line Value Wake Up Condition on UTMIP P2 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
19	0x0	UTMIP_TCTRL_USE_PMC_P2: Use PMC Saved TCTRL on UTMIP P2
18	0x0	UTMIP_PCTRL_USE_PMC_P2: Use PMC Saved PCTRL on UTMIP P2
17	0x0	UTMIP_FSLS_USE_PMC_P2: Use PMC Saved Pad config on UTMIP P2
16	0x0	UTMIP_MASTER_ENABLE_P2: Enable use of master pins on UTMIP P2
15:12	0xc	UTMIP_WAKE_VAL_P1: Line Value Wake Up Condition on UTMIP P1 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
11	0x0	UTMIP_TCTRL_USE_PMC_P1: Use PMC Saved TCTRL on UTMIP P1
10	0x0	UTMIP_PCTRL_USE_PMC_P1: Use PMC Saved PCTRL on UTMIP P1
9	0x0	UTMIP_FSLS_USE_PMC_P1: Use PMC Saved Pad config on UTMIP P1
8	0x0	UTMIP_MASTER_ENABLE_P1: Enable use of master pins on UTMIP P1
7:4	0xc	UTMIP_WAKE_VAL_P0: Line Value Wake Up Condition on UTMIP P0 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY

Bit	Reset	Description
3	0x0	UTMIP_TCTRL_USE_PMC_P0: Use PMC Saved TCTRL on UTMIP P0
2	0x0	UTMIP_PCTRL_USE_PMC_P0: Use PMC Saved PCTRL on UTMIP P0
1	0x0	UTMIP_FSLS_USE_PMC_P0: Use PMC Saved Pad config on UTMIP P0
0	0x0	UTMIP_MASTER_ENABLE_P0: Enable use of master pins on UTMIP P0

12.6.125 APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0

This register determines when sleep walking should take effect: upon a specific GPIO event, on any wake event, or on a line value change.

It is also possible to force a sleep walk; see register UTMIP_UHSIC_TRIGGERS to force the event.

Sleep Walk Sequence Enables

Offset: 0x200 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	UHSIC_LINEVAL_WALK_EN_P0: Perform Walk on USB line value wake up for UHSIC P0
30	0x0	UHSIC_WAKE_WALK_EN_P0: Perform Walk on any chip wake up event for UHSIC P0
29	0x0	UHSIC_GPIO_WALK_EN_P0: Perform Walk on associated GPIO event for UHSIC P0
28:24	0x0	UHSIC_DESIGNATED_GPIO_P0: GPIO Number associated with UHSIC P0
23	0x0	UTMIP_LINEVAL_WALK_EN_P2: Perform Walk on USB line value wake up for UTMIP P2
22	0x0	UTMIP_WAKE_WALK_EN_P2: Perform Walk on any chip wake up event for UTMIP P2
21	0x0	UTMIP_GPIO_WALK_EN_P2: Perform Walk on associated GPIO event for UTMIP P2
20:16	0x0	UTMIP_DESIGNATED_GPIO_P2: GPIO Number associated with UTMIP P2
15	0x0	UTMIP_LINEVAL_WALK_EN_P1: Perform Walk on USB line value wake up for UTMIP P1
14	0x0	UTMIP_WAKE_WALK_EN_P1: Perform Walk on any chip wake up event for UTMIP P1
13	0x0	UTMIP_GPIO_WALK_EN_P1: Perform Walk on associated GPIO event for UTMIP P1
12:8	0x0	UTMIP_DESIGNATED_GPIO_P1: GPIO Number associated with UTMIP P1
7	0x0	UTMIP_LINEVAL_WALK_EN_P0: Perform Walk on USB line value wake up for UTMIP P0
6	0x0	UTMIP_WAKE_WALK_EN_P0: Perform Walk on any chip wake up event for UTMIP P0
5	0x0	UTMIP_GPIO_WALK_EN_P0: Perform Walk on associated GPIO event for UTMIP P0
4:0	0x0	UTMIP_DESIGNATED_GPIO_P0: GPIO Number associated with UTMIP P0

12.6.126 APBDEV_PMC_UTMIP_SLEEPWALK_P0_0

These registers hold the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- MASTER_USBOP_RPD
- MASTER_USBON_RPD
- MASTER_USBOP_RPU
- MASTER_USBON_RPU
- MASTER_AP
- MASTER_AN
- MASTER_HIGHZ

For the walk to take effect at the pad, the MASTER_ENABLE pin must be set high in the config register. Otherwise the pad will ignore the values.

If no walk is enabled or forced, then the walk pointer remains stuck on phase A. The walk pointer should use a 2 bit Gray code so that Phase A is 00, Phase B is 01, Phase C is 11, and Phase D is 10. Once Phase D is reached, only a reset of the phase pointer can bring it back to Phase A.

Four phases should be sufficient to handle most wake-up events.

Signaling Sequence for UTMIP Port 0 Wakeup

Offset: 0x204 | Read/Write: R/W | Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

12.6.127 APBDEV_PMC_UTMIP_SLEEPWALK_P1_0

Signaling Sequence for UTMIP Port 1 Wakeup

Offset: 0x208 | Read/Write: R/W | Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers

Bit	Reset	Description
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

12.6.128 APBDEV_PMC_UTMIP_SLEEPWALK_P2_0

Signaling Sequence for UTMIP Port 2 Wakeup

Offset: 0x20c │ Read/Write: R/W │ Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers, active low
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers, active low
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line

Bit	Reset	Description
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers, active low
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers, active low
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

12.6.129 APBDEV_PMC_UHSIC_SLEEPWALK_P0_0

These registers hold the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- STROBE_RPD
- DATA_RPD
- STROBE_RPU
- DATA_RPU

At this time there are no STROBE, DATA ports that should be driven by the host. So these outputs from the module should be left unconnected until a time that departure may be implemented.

Signaling Sequence for UHSIC Port 0 Wakeup

Offset: 0x210 │ Read/Write: R/W │ Reset: 0x16161616 (0b00010110000101100001011000010110)

Bit	Reset	Description
31:30	0x0	RESERVED_D: Phase D Unused Bit
29	0x0	UHSIC_DATA1_RPU_D: Phase D Pull up on DATA1 Line
28	0x1	UHSIC_DATA1_RPD_D: Phase D Pull Down on DATA1 Line
27	0x0	UHSIC_DATA0_RPU_D: Phase D Pull up on DATA0 Line
26	0x1	UHSIC_STROBE_RPU_D: Phase D Pull Up on STROBE Line
25	0x1	UHSIC_DATA0_RPD_D: Phase D Pull Down on DATA0 Line
24	0x0	UHSIC_STROBE_RPD_D: Phase D Pull Down on STROBE Line
23:22	0x0	RESERVED_C: Phase C Unused Bit
21	0x0	UHSIC_DATA1_RPU_C: Phase C Pull up on DATA1 Line
20	0x1	UHSIC_DATA1_RPD_C: Phase C Pull Down on DATA1 Line

Bit	Reset	Description
19	0x0	UHSIC_DATA0_RPU_C: Phase C Pull up on DATA0 Line
18	0x1	UHSIC_STROBE_RPU_C: Phase C Pull Up on STROBE Line
17	0x1	UHSIC_DATA0_RPD_C: Phase C Pull Down on DATA0 Line
16	0x0	UHSIC_STROBE_RPD_C: Phase C Pull Down on STROBE Line
15:14	0x0	RESERVED_B: Phase B Unused Bit
13	0x0	UHSIC_DATA1_RPU_B: Phase B Pull up on DATA1 Line
12	0x1	UHSIC_DATA1_RPD_B: Phase B Pull Down on DATA1 Line
11	0x0	UHSIC_DATA0_RPU_B: Phase B Pull up on DATA0 Line
10	0x1	UHSIC_STROBE_RPU_B: Phase B Pull Up on STROBE Line
9	0x1	UHSIC_DATA0_RPD_B: Phase B Pull Down on DATA0 Line
8	0x0	UHSIC_STROBE_RPD_B: Phase B Pull Down on STROBE Line
7:6	0x0	RESERVED_A: Phase A Unused Bit
5	0x0	UHSIC_DATA1_RPU_A: Phase A Pull up on DATA1 Line
4	0x1	UHSIC_DATA1_RPD_A: Phase A Pull Down on DATA1 Line
3	0x0	UHSIC_DATA0_RPU_A: Phase A Pull up on DATA0 Line
2	0x1	UHSIC_STROBE_RPU_A: Phase A Pull Up on STROBE Line
1	0x1	UHSIC_DATA0_RPD_A: Phase A Pull Down on DATA0 Line
0	0x0	UHSIC_STROBE_RPD_A: Phase A Pull Down on STROBE Line

12.6.130 APBDEV_PMC_UTMIP_UHSIC_STATUS_0

Read-only register that provides current walk pointer information as well as line value.

Status of UTMIP UHSIC Wake-up Circuitry

Offset: 0x214 │ Read/Write: RO │ Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	DATA1_VAL_P0: Value of DATA1 line detector for UHSIC port 0
24	X	UTMIP_WAKE_ALARM_P3: A wake event occurred on UTMIP port 0
23	X	USBON_VAL_P3: Value of D- line detector for UTMIP port 0
22	X	USBOP_VAL_P3: Value of D+ line detector for UTMIP port 0
21:20	X	UTMIP_WALK_PTR_P3: Walk pointer for UMTIP port 0
19	X	UHSIC_WAKE_ALARM_P0: A wake event occurred on UHSIC port 0
18	X	UTMIP_WAKE_ALARM_P2: A wake event occurred on UTMIP port 2
17	X	UTMIP_WAKE_ALARM_P1: A wake event occurred on UTMIP port 1
16	X	UTMIP_WAKE_ALARM_P0: A wake event occurred on UTMIP port 0
15	X	DATA0_VAL_P0: Value of DATA0 line detector for UHSIC port 0
14	X	STROBE_VAL_P0: Value of STROBE line detector for UHSIC port 0
13	X	USBON_VAL_P2: Value of D- line detector for UTMIP port 2
12	X	USBOP_VAL_P2: Value of D+ line detector for UTMIP port 2
11	X	USBON_VAL_P1: Value of D- line detector for UTMIP port 1
10	X	USBOP_VAL_P1: Value of D+ line detector for UTMIP port 1
9	X	USBON_VAL_P0: Value of D- line detector for UTMIP port 0
8	X	USBOP_VAL_P0: Value of D+ line detector for UTMIP port 0
7:6	X	UHSIC_WALK_PTR_P0: Walk pointer for UHSIC port 0
5:4	X	UTMIP_WALK_PTR_P2: Walk pointer for UTMIP port 2
3:2	X	UTMIP_WALK_PTR_P1: Walk pointer for UTMIP port 1
1:0	X	UTMIP_WALK_PTR_P0: Walk pointer for UMTIP port 0

12.6.131 APBDEV_PMC_UTMIP_UHSIC_FAKE_0

Instead of using the pad value for USBOP_VAL, USBON_VAL, STROBE_VAL, DATA_VAL, VBUS_WAKEUP, ID from the UTMIP and HSIC pads, force a 2x1 mux at the input of the PMC pad macro when fake values are enabled. This could the useful feature of putting a line at rest when needed. It also has important debug merits. This mux should be placed in front of the NP synchronizer for these signals (saved power on the synchronizer). This will have to be waived in our sync checks.

Fake the Line Value for the PMC Pad Macro

Offset: 0x218 | Read/Write: R/W | Reset: 0x01111111 (0bxxxx00010001000100010001000100010001)

Bit	Reset	Description
27	0x0	UTMIP_ID_FAKE_EN_P2: Enable the fake ID value for UTMIP P2
26	0x0	UTMIP_ID_FAKE_VAL_P2: Fake ID value for UTMIP P2
25	0x0	UTMIP_VBUS_FAKE_EN_P2: Enable the fake VBUS WAKEUP value for UTMIP P2
24	0x1	UTMIP_VBUS_FAKE_VAL_P2: Fake VBUS WAKEUP value for UTMIP P2
23	0x0	UTMIP_ID_FAKE_EN_P1: Enable the fake ID value for UTMIP P1
22	0x0	UTMIP_ID_FAKE_VAL_P1: Fake ID value for UTMIP P1
21	0x0	UTMIP_VBUS_FAKE_EN_P1: Enable the fake VBUS WAKEUP value for UTMIP P1
20	0x1	UTMIP_VBUS_FAKE_VAL_P1: Fake VBUS WAKEUP value for UTMIP P1
19	0x0	UTMIP_ID_FAKE_EN_P0: Enable the fake ID value for UTMIP P0
18	0x0	UTMIP_ID_FAKE_VAL_P0: Fake ID value for UTMIP P0
17	0x0	UTMIP_VBUS_FAKE_EN_P0: Enable the fake VBUS WAKEUP value for UTMIP P0
16	0x1	UTMIP_VBUS_FAKE_VAL_P0: Fake VBUS WAKEUP value for UTMIP P0
15	0x0	UHSIC_FAKE_DATA_EN_P0: Enable the fake line value for DATA for the PMC pad macro for UHSIC P0
14	0x0	UHSIC_FAKE_STROBE_EN_P0: Enable the fake line value for STROBE for the PMC pad macro for UHSIC P0
13	0x0	UHSIC_FAKE_DATA_VAL_P0: Fake line value for DATA for the PMC pad macro for UHSIC P0
12	0x1	UHSIC_FAKE_STROBE_VAL_P0: Fake line value for STROBE for the PMC pad macro for UHSIC P0
11	0x0	UTMIP_FAKE_USBON_EN_P2: Enable the fake line value for D- for the PMC pad macro for UTMIP P2
10	0x0	UTMIP_FAKE_USBOP_EN_P2: Enable the fake line value for D+ for the PMC pad macro for UTMIP P2
9	0x0	UTMIP_FAKE_USBON_VAL_P2: Fake line value for D- for the PMC pad macro for UTMIP P2
8	0x1	UTMIP_FAKE_USBOP_VAL_P2: Fake line value for D+ for the PMC pad macro for UTMIP P2
7	0x0	UTMIP_FAKE_USBON_EN_P1: Enable the fake line value for D- for the PMC pad macro for UTMIP P1
6	0x0	UTMIP_FAKE_USBOP_EN_P1: Enable the fake line value for D+ for the PMC pad macro for UTMIP P1
5	0x0	UTMIP_FAKE_USBON_VAL_P1: Fake line value for D- for the PMC pad macro for UTMIP P1
4	0x1	UTMIP_FAKE_USBOP_VAL_P1: Fake line value for D+ for the PMC pad macro for UTMIP P1
3	0x0	UTMIP_FAKE_USBON_EN_P0: Enable the fake line value for D- for the PMC pad macro for UTMIP P0
2	0x0	UTMIP_FAKE_USBOP_EN_P0: Enable the fake line value for D+ for the PMC pad macro for UTMIP P0
1	0x0	UTMIP_FAKE_USBON_VAL_P0: Fake line value for D- for the PMC pad macro for UTMIP P0
0	0x1	UTMIP_FAKE_USBOP_VAL_P0: Fake line value for D+ for the PMC pad macro for UTMIP P0

12.6.132 APBDEV_PMC_BONDOUT_MIRROR3_0

Secure Scratch Register

Offset: 0x21c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR3

12.6.133 APBDEV_PMC_BONDOUT_MIRROR4_0

Secure Scratch Register

Offset: 0x220 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR4

12.6.134 APBDEV_PMC_SECURE_SCRATCH6_0

Secure Scratch Register

Offset: 0x224 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH6

12.6.135 APBDEV_PMC_SECURE_SCRATCH7_0

Secure Scratch Register

Offset: 0x228 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH7

12.6.136 APBDEV_PMC_SCRATCH43_0

Scratch Register

Offset: 0x22c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH43: General-purpose register storage. Currently used for power-gating progress status in RTAPI.

12.6.137 APBDEV_PMC_SCRATCH44_0

Scratch Register

Offset: 0x230 | Read/Write: R/W V Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH44: General-purpose register storage. Currently used for storing ARM SP in power-gating by RTAPI.

12.6.138 APBDEV_PMC_SCRATCH45_0

Scratch Register

Offset: 0x234 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH45: General-purpose register storage

12.6.139 APBDEV_PMC_SCRATCH46_0

Scratch Register

Offset: 0x238 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH46: General-purpose register storage

12.6.140 APBDEV_PMC_SCRATCH47_0

Scratch Register

Offset: 0x23c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH47: General-purpose register storage

12.6.141 APBDEV_PMC_SCRATCH48_0

Scratch Register

Offset: 0x240 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH48: General-purpose register storage

12.6.142 APBDEV_PMC_SCRATCH49_0

Scratch Register

Offset: 0x244 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH49: General-purpose register storage

12.6.143 APBDEV_PMC_SCRATCH50_0

Scratch Register

Offset: 0x248 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH50: General-purpose register storage

12.6.144 APBDEV_PMC_SCRATCH51_0

Scratch Register

Offset: 0x24c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH51: General-purpose register storage

12.6.145 APBDEV_PMC_SCRATCH52_0

Scratch Register

This register is valid in I2C mode. It is used for the WATCHDOG_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x250 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SCRATCH_PMU_A_0_HIWORD_RANGE
15:0	X	SCRATCH_PMU_A_0_LOWORD_RANGE

12.6.146 APBDEV_PMC_SCRATCH53_0

Scratch Register

This register is valid in I2C mode. It is used for the WATCHDOG_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x254 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SCRATCH_PMU_B_0_RST_ENABLE_RANGE: Reset enable.
30	X	SCRATCH_PMU_B_0_CNTLR_TYPE_RANGE: This bit is reserved. Set it to 0.
29:27	X	SCRATCH_PMU_B_0_CNTLR_ID_RANGE: Controller ID. 0: I2C1 1: I2C2 2: I2C3 3: I2C4 4: I2C PMU
26:24	X	SCRATCH_PMU_B_0_PINMUX_RANGE. Pinmux Range. Refer to Section 13.6.2: Pinmux Support in the Boot Process chapter for the encoding of this field.
23:16	X	SCRATCH_PMU_B_0_CHKSUM_RANGE: Checksum Range. Every byte in this register must add to 0. Refer to Section 13.6.3: Checksum Calculation in the Boot Process section for the encoding of this field.
15	X	SCRATCH_PMU_B_0_16BITOP_RANGE: 16-Bit Op.
14	X	SCRATCH_PMU_B_0_USE_GPIO_RANGE. Set this bit to 0.
13:7	X	Reserved. Set these bits to 0.
6:0	X	SCRATCH_PMU_B_0_I2CSLV1_RANGE: I2C Slave Address.

12.6.147 APBDEV_PMC_SCRATCH54_0

Scratch Register

This register is valid in I2C mode. It is used for the TSENSE_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x258 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SCRATCH_PMU_A_0_HIWORD_RANGE

Bit	Reset	Description
15:0	X	SCRATCH_PMU_A_0_LOWORD_RANGE

12.6.148 APBDEV_PMC_SCRATCH55_0

Scratch Register

This register is valid in I2C mode. It is used for the TSENSE_RESET I2C command. In GPIO and SPI modes, this register is reserved.

Offset: 0x25c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SCRATCH_PMU_B_0_RST_ENABLE_RANGE: Reset enable.
30	X	SCRATCH_PMU_B_0_CNTLR_TYPE_RANGE: This bit is reserved. Set it to 0.
29:27	X	SCRATCH_PMU_B_0_CNTLR_ID_RANGE: Controller ID. 0: I2C1 1: I2C2 2: I2C3 3: I2C4 4: I2C PMU
26:24	X	SCRATCH_PMU_B_0_PINMUX_RANGE. Pinmux Range. Refer to Section 13.6.2: Pinmux Support" in the Boot Process chapter for the encoding of this field.
23:16	X	SCRATCH_PMU_B_0_CHKSUM_RANGE: Checksum Range. Every byte in this register must add to 0. Refer to Section 13.6.3: Checksum Calculation in the Boot Process chapter for the encoding of this field.
15	X	SCRATCH_PMU_B_0_16BITOP_RANGE: 16-Bit Op.
14	X	SCRATCH_PMU_B_0_USE_GPIO_RANGE. Set this bit to 0.
13:7	X	Reserved. Set these bits to 0.
6:0	X	SCRATCH_PMU_B_0_I2CSLV1_RANGE: I2C Slave Address.

12.6.149 APBDEV_PMC_SCRATCH0_ECO_0

Scratch Register

Offset: 0x260 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CAR_USE_NEW_LP0_EXIT_SEQ: Reserved.
30:0	0x0	ECO0: Reserved.

12.6.150 APBDEV_PMC_POR_DPD_CTRL_0

This register is defunct.

Offset: 0x264 | Read/Write: R/W | Reset: 0x80000003 (0b1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
31	0x1	MEM0_HOLD_CKE_LOW_OVR: DEFUNCT
1	0x1	MEM0_ADDR1_CLK_SEL_DPD: DEFUNCT
0	0x1	MEM0_ADDR0_CLK_SEL_DPD: DEFUNCT

12.6.151 APBDEV_PMC_SCRATCH2_ECO_0

Scratch Register

Offset: 0x268 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	ECO2: Reserved.

12.6.152 APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0

Offset: 0x26c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4	0x1	UTMIP_LINE_WAKEUP_EN_P3: enables latching line wakeup event on UTMIP p3
3	0x1	UHSIC_LINE_WAKEUP_EN_P0: enables latching line wakeup event on UHSIC p0
2	0x1	UTMIP_LINE_WAKEUP_EN_P2: enables latching line wakeup event on UTMIP p2
1	0x1	UTMIP_LINE_WAKEUP_EN_P1: enables latching line wakeup event on UTMIP p1
0	0x1	UTMIP_LINE_WAKEUP_EN_P0: enables latching line wakeup event on UTMIP p0

12.6.153 APBDEV_PMC_UTMIP_BIAS_MASTER_CNTRL_0

Offset: 0x270 | Read/Write: R/W | Reset: 0x0000000d (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1101)

Bit	Reset	Description
3	0x1	UTMIP_BIAS_MASTER_AWAKE_AND: When not PROGRAMMABLE and E_DPD=0, AND the MASTER_ENABLE of all IO ports for bias cell. If not AND or OR, force to 0.
2	0x1	UTMIP_BIAS_MASTER_AWAKE_OR: When not PROGRAMMABLE and E_DPD=0, OR the MASTER_ENABLE of all IO ports for bias cell
1	0x0	UTMIP_BIAS_MASTER_PROG_VAL: When PROGRAMMABLE, this is the value to program too
0	0x1	UTMIP_BIAS_MASTER_PROG_CTRL: MASTER_ENABLE pin uses a programmable value on BIAS pad, at all times

12.6.154 APBDEV_PMC_UTMIP_MASTER_CONFIG_0

MASTER_CONFIG selects the MASTER function's mode of operation, in this case:

- 0 for 500 µA usage
- 1 for smaller current usage when driving.

Offset: 0x274 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	UTMIP_PWR_P3: enables UTMIP p3 low power mode
3	0x0	UHSIC_PWR_P0: enables UHSIC p0 low power mode
2	0x0	UTMIP_PWR_P2: enables UTMIP p2 low power mode
1	0x0	UTMIP_PWR_P1: enables UTMIP p1 low power mode
0	0x0	UTMIP_PWR_P0: enables UTMIP p0 low power mode

12.6.155 APBDEV_PMC_TD_PWRGATE_INTER_PART_TIMER_0

TD Power-Gating Timing Between Partition Power-Gating

Offset: 0x278 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	DATA: timing between consecutive partitions

12.6.156 APBDEV_PMC_UTMIP_UHSIC2_TRIGGERS_0

PMC trigger events for the UTMIP ports as well as the UHSIC port. Sleep walk clearing returns the walk pointer to 00, the first entry in a USB line walk. This is the only way to return a pointer to 0.

Pad configuration capture a set of FLLS parameters prior to resting the port so their value can be retained in deep sleep.

The trigger should only happen on the fields that are set to TRIG (1) when writing. Returned read value should always be all NULL fields. The FORCE_WALK bits trigger a SLEEP

This does not need to be a real register, so it should not consume much area. Any value read back should be 0.

Triggers for USB Ports

Offset: 0x27c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	UHSIC_CLR_WAKE_ALARM_P1: Clear wake event for UHSIC port 0 0 = NULL 1 = TRIG
2	X	UHSIC_FORCE_WALK_P1: Force pointer walk for UHSIC port 0 0 = NULL 1 = TRIG
1	X	UHSIC_RESERVED_P1: Reserved for UHSIC port 0 0 = NULL 1 = TRIG
0	X	UHSIC_CLR_WALK_PTR_P1: Clear sleep walk pointer for UHSIC port 1 0 = NULL 1 = TRIG

12.6.157 APBDEV_PMC_UTMIP_UHSIC2_SAVED_STATE_0

Save some critical information about USB port prior to entering DPD in a suspend state.

SPEED:

- HS: Suspend DPD was entered from a high speed mode, restore high speed at DPD exit
- FS: Suspend DPD was entered from a full speed mode, restore full speed at DPD exit
- LS: Suspend DPD was entered from a low speed mode, restore low speed at DPD exit
- RST: Reset Port, Hardware reset or DPD was not entered suspended bus, Reset and Re-enumerate port

SCRATCH -- Save other critical information about the port, software choice. Reset value should read 0x0F0F0F0F, reset should occur on Cold Hardware Reset only

Saved DPD State for all UTMIP and UHSIC Ports

Offset: 0x280 │ Read/Write: R/W │ Reset: 0x00000f07 (0bxxxxxxxxxxxxxxxx0000111100000111)

Bit	Reset	Description
15	0x0	UTMIP_WAKE_EX_P3:wakeup on anything except a Particular Line Value 0 = OFF 1 = ON
14	0x0	UTMIP_IGNORE_MASTER_CFG_P3
13:10	0x3	UTMIP_SCRATCH_P3
9:8	0x3	UTMIP_SPEED_P3: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST
7	0x0	UHSIC_WAKE_EX_P1: Wake up on anything except a particular line value 0 = OFF 1 = ON

Bit	Reset	Description
6	0x0	UHSIC_IGNORE_MASTER_CFG_P1
5:1	0x3	UHSIC_SCRATCH_P1
0	0x1	UHSIC_MODE_P1: UHSIC Speed prior to DPD 0 = HS 1 = RST

12.6.158 APBDEV_PMC_UTMIP_UHSIC2_SLEEP_CFG_0

This register configures the value of the line that could cause a wake-up event.

- ANY: signifies any line change DM EDGE or DP EDGE for UTMIP.
- NONE: signifies no wake-up possible.

One can use the two most significant bits (3:2) of the WAKE_VAL to 4x1 select whether:

- 00: Check both bits for value on two LSB (1:0).
- 01: Check bit DM (or strobe for HSIC) against bit 0
- 10: Check bit DP (or DATA for HSIC) against bit 1
- 11: Check for edge on DP against bit 1, edge on DM against bit 0

Sleep Walk Sequence Enables

Offset: 0x284 | Read/Write: R/W | Reset: 0x000000c0 (0bxxxxxxxxxxxxxxxxxxxxxxxxx11000000)

Bit	Reset	Description
7:4	0xc	UHSIC_WAKE_VAL_P1: Line Value Wake Up Condition on UHSIC P1 0 = SD00 1 = SD01 2 = SD10 3 = SD11 4 = S0 5 = S1 8 = D0 10 = D1 12 = NONE 13 = SEDGE 14 = DEDGE 15 = ANY
3:1	0x0	UHSIC_RESERVED_P1: Reserved for later use for UHSIC P1
0	0x0	UHSIC_MASTER_ENABLE_P1: Enable use of master pins on UHSIC P1

12.6.159 APBDEV_PMC_UTMIP_UHSIC2_SLEEPWALK_CFG_0

This register determines when sleep walking should take effect: upon a specific GPIO event, on any wake event, or on a line value change.

> ***Note:*** *It is also possible to force a sleep walk: see register UTMIP_UHSIC_TRIGGERS to force the event.*

Sleep Walk Sequence Enables

Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	UTMIP_LINEVAL_WALK_EN_P3: Perform Walk on USB line value wake up for UTMIP P3
14	0x0	UTMIP_WAKE_WALK_EN_P3: Perform Walk on any chip wake up event for UTMIP P3
13	0x0	UTMIP_GPIO_WALK_EN_P3: Perform Walk on associated GPIO event for UTMIP P3

Bit	Reset	Description
12:8	0x0	UTMIP_DESIGNATED_GPIO_P3: GPIO Number associated with UTMIP P3
7	0x0	UHSIC_LINEVAL_WALK_EN_P1: Perform walk on USB line value wake up for UHSIC P1
6	0x0	UHSIC_WAKE_WALK_EN_P1: Perform walk on any chip wake up event for UHSIC P1
5	0x0	UHSIC_GPIO_WALK_EN_P1: Perform walk on associated GPIO event for UHSIC P1
4:0	0x0	UHSIC_DESIGNATED_GPIO_P1: GPIO Number associated with UHSIC P1

12.6.160 APBDEV_PMC_UHSIC2_SLEEPWALK_P1_0

This register holds the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- STROBE_RPD
- DATA_RPD
- STROBE_RPU
- DATA_RPU

At this time there are no STROBE, DATA ports that should be driven by the host. So these outputs from the module should be left dangling (unconnected) until a time that feature may be implemented.

Signaling Sequence for UHSIC Port 0 Wakeup

Offset: 0x28c │ Read/Write: R/W │ Reset: 0x06060606 (0b00000110000001100000011000000110)

Bit	Reset	Description
31:28	0x0	RESERVED_D: Phase D Unused Bit
27	0x0	UHSIC_DATA_RPU_D: Phase D Pull up on DATA Line
26	0x1	UHSIC_STROBE_RPU_D: Phase D Pull Up on STROBE Line
25	0x1	UHSIC_DATA_RPD_D: Phase D Pull Down on DATA Line
24	0x0	UHSIC_STROBE_RPD_D: Phase D Pull Down on STROBE Line
23:20	0x0	RESERVED_C: Phase C Unused Bit
19	0x0	UHSIC_DATA_RPU_C: Phase C Pull up on DATA Line
18	0x1	UHSIC_STROBE_RPU_C: Phase C Pull Up on STROBE Line
17	0x1	UHSIC_DATA_RPD_C: Phase C Pull Down on DATA Line
16	0x0	UHSIC_STROBE_RPD_C: Phase C Pull Down on STROBE Line
15:12	0x0	RESERVED_B: Phase B Unused Bit
11	0x0	UHSIC_DATA_RPU_B: Phase B Pull up on DATA Line
10	0x1	UHSIC_STROBE_RPU_B: Phase B Pull Up on STROBE Line
9	0x1	UHSIC_DATA_RPD_B: Phase B Pull Down on DATA Line
8	0x0	UHSIC_STROBE_RPD_B: Phase B Pull Down on STROBE Line
7:4	0x0	RESERVED_A: Phase A Unused Bit
3	0x0	UHSIC_DATA_RPU_A: Phase A Pull up on DATA Line
2	0x1	UHSIC_STROBE_RPU_A: Phase A Pull Up on STROBE Line
1	0x1	UHSIC_DATA_RPD_A: Phase A Pull Down on DATA Line
0	0x0	UHSIC_STROBE_RPD_A: Phase A Pull Down on STROBE Line

12.6.161 APBDEV_PMC_UTMIP_UHSIC2_STATUS_0

Read-only register that provides current walk pointer information as well as the line value.

Status of UTMIP UHSIC Wakeup Circuitry

Offset: 0x290 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	UHSIC_WAKE_ALARM_P1: A wake event occurred on UHSIC port 1
3	X	DATA_VAL_P1: Value of DATA line detector for UHSIC port 1
2	X	STROBE_VAL_P1: Value of STROBE line detector for UHSIC port 1
1:0	X	UHSIC_WALK_PTR_P1: Walk pointer for UHSIC port 0

12.6.162 APBDEV_PMC_UTMIP_UHSIC2_FAKE_0

Instead of using the pad value for USBOP_VAL, USBON_VAL, STROBE_VAL, DATA_VAL, VBUS_WAKEUP, ID from the UTMIP and HSIC pads, force a 2x1 mux at the input of the PMC pad macro when fake values are enabled. This could the useful feature of putting a line at rest when needed. It also has important debug merits. This mux should be placed in front of the NP synchronizer for these signals (saved power on the synchronizer). This will have to be waived in our sync checks.

Fake the Line Value for the PMC Pad Macro

Offset: 0x294 | Read/Write: R/W | Reset: 0x00001101 (0bxxxxxxxxxxxxxxxx0001000100000001)

Bit	Reset	Description
15	0x0	UTMIP_ID_FAKE_EN_P3: Enable the fake ID value for UTMIP P2
14	0x0	UTMIP_ID_FAKE_VAL_P3: Fake ID value for UTMIP P2
13	0x0	UTMIP_VBUS_FAKE_EN_P3: Enable the fake VBUS WAKEUP value for UTMIP P2
12	0x1	UTMIP_VBUS_FAKE_VAL_P3: Fake VBUS WAKEUP value for UTMIP P2
11	0x0	UTMIP_FAKE_USBON_EN_P3: Enable the fake line value for D- for the PMC pad macro for UTMIP P0
10	0x0	UTMIP_FAKE_USBOP_EN_P3: Enable the fake line value for D+ for the PMC pad macro for UTMIP P0
9	0x0	UTMIP_FAKE_USBON_VAL_P3: Fake line value for D- for the PMC pad macro for UTMIP P0
8	0x1	UTMIP_FAKE_USBOP_VAL_P3: Fake line value for D+ for the PMC pad macro for UTMIP P0
7:4	0x0	RESERVED
3	0x0	UHSIC_FAKE_DATA_EN_P1: Enable the fake line value for DATA for the PMC pad macro for UHSIC P0
2	0x0	UHSIC_FAKE_STROBE_EN_P1: Enable the fake line value for STROBE for the PMC pad macro for UHSIC P0
1	0x0	UHSIC_FAKE_DATA_VAL_P1: Fake line value for DATA for the PMC pad macro for UHSIC P0
0	0x1	UHSIC_FAKE_STROBE_VAL_P1: Fake line value for STROBE for the PMC pad macro for UHSIC P0

12.6.163 APBDEV_PMC_UTMIP_UHSIC2_LINE_WAKEUP_0

Offset: 0x298 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	UHSIC_LINE_WAKEUP_EN_P1: enables latching line wake-up event on UHSIC P1

12.6.164 APBDEV_PMC_UTMIP_MASTER2_CONFIG_0

MASTER_CONFIG selects the MASTER functions mode of operation, in this case 0 for 500uA usage, 1 for smaller current usage when driving.

Offset: 0x29c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UHSIC_PWR_P1: enables UHSIC P1 low power mode

12.6.165 APBDEV_PMC_UTMIP_UHSIC_RPD_CFG_0

Offset: 0x2a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	WEAKPD_ANYTIME_P3
10	0x0	DP_WEAKPD_CFG_P3
9	0x0	DM_WEAKPD_CFG_P3
8	0x0	WEAKPD_ANYTIME_P2
7	0x0	DP_WEAKPD_CFG_P2
6	0x0	DM_WEAKPD_CFG_P2
5	0x0	WEAKPD_ANYTIME_P1
4	0x0	DP_WEAKPD_CFG_P1
3	0x0	DM_WEAKPD_CFG_P1
2	0x0	WEAKPD_ANYTIME_P0
1	0x0	DP_WEAKPD_CFG_P0
0	0x0	DM_WEAKPD_CFG_P0

12.6.166 APBDEV_PMC_PG_MASK_CE0_0

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

12.6.167 APBDEV_PMC_PG_MASK_3_0

Offset: 0x2a8 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	DISB: Mask DISB rail
23:16	0xff	DIS: Mask DIS rail
15:8	0xff	RESERVED_C1NC: Reserved
7:0	0xff	C0NC: Mask C0NC rail

12.6.168 APBDEV_PMC_PG_MASK_4_0

Offset: 0x2ac | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	SOR: Mask SOR rail
23:16	0xff	XUSBC: Mask XUSBC rail
15:8	0xff	XUSBB: Mask XUSBB rail
7:0	0xff	XUSBA: Mask XUSBA rail

12.6.169 APBDEV_PMC_PLLM_WB0_OVERRIDE2_0

Offset: 0x2b0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxx0000000000000000)

Bit	Reset	Description
31:27	0x0	DIVP: 4 bit DIVP for PLLM
26	0x0	KVCO: KVCO/VCO gain
25:24	0x0	KCP: Charge pump control
15:0	0x0	SETUP: Setup

12.6.170 APBDEV_PMC_TSC_MULT_0

Offset: 0x2b4 | Read/Write: R/W | Reset: 0x000016e0 (0bxxxxxxxxxxxxx000x0001011011100000)

Bit	R/W	Reset	Description
19:17	RW	0x0	TICK_SEL: This bit selects one of the six binary time stamp counter bits. 0 = BIT0 1 = BIT1 2 = BIT2 3 = BIT3 4 = BIT4 5 = BIT5
16	RO	X	FREQ_STS: Clock frequency being used for counter. 0 = Fast (i.e., oscillator frequency). 1 = Slow (i.e., 32 kHz). When the PMC_DPD_ENABLE[TSC_MULT_EN] bit is set to '1', the PMC waits for first rising edge on the slow clock before switching the counter to the slow clock. This status is also set at the first rising edge of the slow clock (to indicate the change of clock for counter).
15:0	RW	0x16e0	MULT_VAL: TSC multiply value (default based on 12 MHz oscillator).Value is (osc-freq * 16 / 32.768 kHz) when the oscillator is disabled and the TSC runs at the 32 kHz clock. For example, for a 12 MHz oscillator, VALUE = 5856.

12.6.171 APBDEV_PMC_CPU_VSENSE_OVERRIDE_0

Offset: 0x2b8 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5	0x0	VDD: VDD sensing override.
4	0x1	C0NC: C0NC VVDD partition sensing override.
3	0x1	CE3:CE3 VVDD partition sensing override.
2	0x1	CE2:CE2 VVDD partition sensing override.
1	0x1	CE1:CE1 VVDD partition sensing override.
0	0x1	CE0:CE0 VVDD partition sensing override.

12.6.172 APBDEV_PMC_GLB_AMAP_CFG_0

This register configures some of the address apertures (in CCPLEX-AXD) to be MMIO or DRAM. Each bit is used to configurable one of the address aperture. For each bit 0=>MMIO and 1=>DRAM.

By default, all configurable apertures are MMIO address space. But they can be configured (at boot time) to be DRAM by programming this register. This register can be write-disabled (by the PMC_SEC_DISABLE register). The bits of this register are used as follows.

GLB_AMAP_CFG

Offset: 0x2bc | Read/Write: R/W | Reset: 0x00020000 (0bxxxxxxxxxxxxxxx10000000000000000)

Bit	Reset	Description
17	0x1	IROM_HIVEC: Defunct. Always DRAM. 0 = MMIO 1 = DRAM
16	0x0	AHB_A2_RSVD aperture: 0 = MMIO 1 = DRAM
15	0x0	AHB_A1_RSVD aperture: 0 = MMIO 1 = DRAM
14	0x0	AHB_A1 aperture: 0 = MMIO 1 = DRAM
13	0x0	APB_RSVD aperture: 0 = MMIO 1 = DRAM
12	0x0	EXTIO_RSVD aperture: 0 = MMIO 1 = DRAM
11	0x0	PPSB_RSVD aperture: 0 = MMIO 1 = DRAM
10	0x0	GART_GPU: Defunct. 0 = MMIO 1 = DRAM
9	0x0	GFX_HOST_RSVD aperture: 0 = MMIO 1 = DRAM
8	0x0	VERIF_RSVD aperture: 0 = MMIO 1 = DRAM
7	0x0	NOR_A3 aperture: 0 = MMIO 1 = DRAM
6	0x0	NOR_A2 aperture: 0 = MMIO 1 = DRAM
5	0x0	NOR_A1 aperture: 0 = MMIO 1 = DRAM
4	0x0	IRAM_RSVD aperture: 0 = MMIO 1 = DRAM
3	0x0	PCIE_A3 aperture: 0 = MMIO 1 = DRAM
2	0x0	PCIE_A2 aperture: 0 = MMIO 1 = DRAM
1	0x0	PCIE_A1 aperture: 0 = MMIO 1 = DRAM
0	0x0	IROM_LOVEC aperture: 0 = MMIO 1 = DRAM

12.6.173 APBDEV_PMC_STICKY_BITS_0

Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000xxxxxxx0)

Bit	Reset	Description
31:9	0x0	RESERVED_1: Reserved
8	0x0	VI: This bit is set to '1' only by a write of '1'. All other writes are ignored. Once it is set to '1', it remains '1' and only resets by a system reset (same reset that resets the main PMC).
0	0x0	HDA_LPBK_DIS: Sticky one bit to disable the loopback in HDA codec.

12.6.174 APBDEV_PMC_SEC_DISABLE2_0

Offset: 0x2c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ23: disable reads from secure register 23 0 = OFF 1 = ON
30	0x0	WRITE23: disable writes to secure register 23 0 = OFF 1 = ON
29	0x0	READ22: disable reads from secure register 22 0 = OFF 1 = ON
28	0x0	WRITE22: disable writes to secure register 22 0 = OFF 1 = ON
27	0x0	READ21: disable reads from secure register 21 0 = OFF 1 = ON
26	0x0	WRITE21: disable writes to secure register 21 0 = OFF 1 = ON
25	0x0	READ20: disable reads from secure register 20 0 = OFF 1 = ON
24	0x0	WRITE20: disable writes to secure register 20 0 = OFF 1 = ON
23	0x0	READ19: disable reads from secure register 19 0 = OFF 1 = ON
22	0x0	WRITE19: disable writes to secure register 19 0 = OFF 1 = ON
21	0x0	READ18: disable reads from secure register 18 0 = OFF 1 = ON
20	0x0	WRITE18: disable writes to secure register 18 0 = OFF 1 = ON
19	0x0	READ17: disable reads from secure register 17 0 = OFF 1 = ON
18	0x0	WRITE17: disable writes to secure register 17 0 = OFF 1 = ON
17	0x0	READ16: disable reads from secure register 16 0 = OFF 1 = ON

Bit	Reset	Description
16	0x0	WRITE16: disable writes to secure register 16 0 = OFF 1 = ON
15	0x0	READ15: disable reads from secure register 15 0 = OFF 1 = ON
14	0x0	WRITE15: disable writes to secure register 15 0 = OFF 1 = ON
13	0x0	READ14: disable reads from secure register 14 0 = OFF 1 = ON
12	0x0	WRITE14: disable writes to secure register 14 0 = OFF 1 = ON
11	0x0	READ13: disable reads from secure register 13 0 = OFF 1 = ON
10	0x0	WRITE13: disable writes to secure register 13 0 = OFF 1 = ON
9	0x0	READ12: disable reads from secure register 12 0 = OFF 1 = ON
8	0x0	WRITE12: disable writes to secure register 12 0 = OFF 1 = ON
7	0x0	READ11: disable reads from secure register 11 0 = OFF 1 = ON
6	0x0	WRITE11: disable writes to secure register 11 0 = OFF 1 = ON
5	0x0	READ10: disable reads from secure register 10 0 = OFF 1 = ON
4	0x0	WRITE10: disable writes to secure register 10 0 = OFF 1 = ON
3	0x0	READ9: disable reads from secure register 9 0 = OFF 1 = ON
2	0x0	WRITE9: disable writes to secure register 9 0 = OFF 1 = ON
1	0x0	READ8: disable reads from secure register 8 0 = OFF 1 = ON
0	0x0	WRITE8: disable writes to secure register 8 0 = OFF 1 = ON

12.6.175 APBDEV_PMC_WEAK_BIAS_0

Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	VTT_E_WB_DDLL: weak bias enable for ddll pad
29	0x0	VTT_E_WB_BR11: weak bias enable for brick11
28	0x0	VTT_E_WB_BR10: weak bias enable for brick10
27	0x0	VTT_E_WB_BR9: weak bias enable for brick9

Bit	Reset	Description
26	0x0	VTT_E_WB_BR8: weak bias enable for brick8
25	0x0	VTT_E_WB_BR7: weak bias enable for brick7
24	0x0	VTT_E_WB_BR6: weak bias enable for brick6
23	0x0	VTT_E_WB_BR5: weak bias enable for brick5
22	0x0	VTT_E_WB_BR4: weak bias enable for brick4
21	0x0	VTT_E_WB_BR3: weak bias enable for brick3
20	0x0	VTT_E_WB_BR2: weak bias enable for brick2
19	0x0	VTT_E_WB_BR1: weak bias enable for brick1
18	0x0	VTT_E_WB_BR0: weak bias enable for brick0
17	0x0	VTT_E_WB_CDBBUF_1: weak bias enable for cdbbuf_1 pad
16	0x0	VTT_E_WB_CDBBUF_0: weak bias enable for cdbbuf_0 pad
15:14	0x0	XM0_ADDR1: Defunct
13:12	0x0	XM0_CLK_B: Defunct
11:10	0x0	XM0_DATA0: Defunct
9:8	0x0	EMMC: Defunct
7:6	0x0	XM1_EXCLK: Defunct
5:4	0x0	XM1_CLK: Defunct
3:2	0x0	XM0_ADDR0: Defunct
1:0	0x0	XM0_CLK: Defunct

12.6.176 APBDEV_PMC_REG_SHORT_0

This register is defunct.

Offset: 0x2cc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19	0x0	DPD_VAL_XM1_DATA: Defunct
18	0x0	DPD_VAL_XM1_ADDR1: Defunct
17	0x0	DPD_VAL_XM0_DATA1: Defunct
16	0x0	DPD_VAL_XM0_ADDR1: Defunct
15	0x0	VAL_XM1_DATA: Defunct
14	0x0	VAL_XM1_ADDR1: Defunct
13	0x0	VAL_XM0_DATA1: Defunct
12	0x0	VAL_XM0_ADDR1: Defunct
11	0x0	DPD_VAL_XM0_CLK_B: Defunct
10	0x0	DPD_VAL_XM0_DATA0: Defunct
9	0x0	VAL_XM0_CLK_B: Defunct
8	0x0	VAL_XM0_DATA0: Defunct
7	0x0	DPD_VAL_XM1_EXCLK: Defunct
6	0x0	DPD_VAL_XM1_CLK: Defunct
5	0x0	DPD_VAL_XM0_ADDR0: Defunct
4	0x0	DPD_VAL_XM0_CLK: Defunct
3	0x0	VAL_XM1_EXCLK: Defunct
2	0x0	VAL_XM1_CLK: Defunct
1	0x0	VAL_XM0__ADDR0: Defunct
0	0x0	VAL_XM0_CLK: Defunct

12.6.177 APBDEV_PMC_PG_MASK_ANDOR_0

Power Gate Mask AND (for force power-gating) or OR (for force power-ungating) the mask value comes from the corresponding PG_MASK* register

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx0000x0xx00x0)

Bit	Reset	Description
29	0x0	VE2: 0 = AND 1 = OR
28	0x0	DFD: 0 = AND 1 = OR
27	0x0	AUD: 0 = AND 1 = OR
26	0x0	NVJPG: 0 = AND 1 = OR
25	0x0	NVDEC: 0 = AND 1 = OR
24	0x0	IRAM: 0 = AND 1 = OR
23	0x0	VIC: 0 = AND 1 = OR
22	0x0	XUSBC: Used as AND or OR override for XUSBC. 0 = AND 1 = OR
21	0x0	XUSBB: Used as AND or OR override for XUSBB. 0 = AND 1 = OR
20	0x0	XUSBA: Used as AND or OR override for XUSBA. 0 = AND 1 = OR
19	0x0	DISB: Used as AND or OR override for DISB partition. 0 = AND 1 = OR
18	0x0	DIS: Used as AND or OR override for DIS partition. 0 = AND 1 = OR
17	0x0	SOR: 0 = AND 1 = OR
16	0x0	C1NC: Used as AND or OR override for Cluster 1 Non-CPU partition. 0 = AND 1 = OR
15	0x0	C0NC: Used as AND or OR override for Cluster 0 Non-CPU partition. 0 = AND 1 = OR
14	0x0	CE0: Used as AND or OR override for CPU0 partition. 0 = AND 1 = OR
11	0x0	CE3: Used as AND or OR override for CE3 partition. 0 = AND 1 = OR
10	0x0	CE2: Used as AND or OR override for CE2 partition. 0 = AND 1 = OR

Bit	Reset	Description
9	0x0	CE1: Used as AND or OR override for CE1 partition. 0 = AND 1 = OR
8	0x0	SAX: 0 = AND 1 = OR
6	0x0	MPE: NV_ADDRESS_MAP_PWRGATE_HEG 0 = AND 1 = OR
3	0x0	PCX: 0 = AND 1 = OR
2	0x0	VE: Used as AND or OR override for VE partition. 0 = AND 1 = OR
0	0x0	RESERVED: NV_ADDRESS_MAP_PWRGATE_TD 0 = AND 1 = OR

12.6.178 APBDEV_PMC_GPU_RG_CNTRL_0

GPU Rail Gating Register

Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	RAIL_CLAMP: Enabling and removing GPU-SOC clamps 0 = DISABLE 1 = ENABLE

12.6.179 APBDEV_PMC_SEC_DISABLE3_0

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ39: disable read from secure register 39 0 = OFF 1 = ON
30	0x0	WRITE39: disable write to secure register 39 0 = OFF 1 = ON
29	0x0	READ38: disable read from secure register 38 0 = OFF 1 = ON
28	0x0	WRITE38: disable write to secure register 38 0 = OFF 1 = ON
27	0x0	READ37: disable read from secure register 37 0 = OFF 1 = ON
26	0x0	WRITE37: disable write to secure register 37 0 = OFF 1 = ON
25	0x0	READ36: disable read from secure register 36 0 = OFF 1 = ON
24	0x0	WRITE36: disable write to secure register 36 0 = OFF 1 = ON
23	0x0	READ35: disable reads from secure register 35 0 = OFF 1 = ON

Bit	Reset	Description
22	0x0	WRITE35: disable writes to secure register 35 0 = OFF 1 = ON
21	0x0	READ34: disable reads from secure register 34 0 = OFF 1 = ON
20	0x0	WRITE34: disable writes to secure register 34 0 = OFF 1 = ON
19	0x0	READ33: disable reads from secure register 33 0 = OFF 1 = ON
18	0x0	WRITE33: disable writes to secure register 33 0 = OFF 1 = ON
17	0x0	READ32: disable reads from secure register 32 0 = OFF 1 = ON
16	0x0	WRITE32: disable writes to secure register 32 0 = OFF 1 = ON
15	0x0	READ31: disable reads from secure register 31 0 = OFF 1 = ON
14	0x0	WRITE31: disable writes to secure register 31 0 = OFF 1 = ON
13	0x0	READ30: disable reads from secure register 30 0 = OFF 1 = ON
12	0x0	WRITE30: disable writes to secure register 30 0 = OFF 1 = ON
11	0x0	READ29: disable reads from secure register 29 0 = OFF 1 = ON
10	0x0	WRITE29: disable writes to secure register 29 0 = OFF 1 = ON
9	0x0	READ28: disable reads from secure register 28 0 = OFF 1 = ON
8	0x0	WRITE28: disable writes to secure register 28 0 = OFF 1 = ON
7	0x0	READ27: disable reads from secure register 27 0 = OFF 1 = ON
6	0x0	WRITE27: disable writes to secure register 27 0 = OFF 1 = ON
5	0x0	READ26: disable reads from secure register 26 0 = OFF 1 = ON
4	0x0	WRITE26: disable writes to secure register 26 0 = OFF 1 = ON
3	0x0	READ25: disable reads from secure register 25 0 = OFF 1 = ON

Bit	Reset	Description
2	0x0	WRITE25: disable writes to secure register 25 0 = OFF 1 = ON
1	0x0	READ24: disable reads from secure register 24 0 = OFF 1 = ON
0	0x0	WRITE24: disable writes to secure register 24 0 = OFF 1 = ON

12.6.180 APBDEV_PMC_PG_MASK_5_0

Offset: 0x2dc | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	DFD: Mask DFD rail
23:16	0xff	AUD: Mask AUD rail
15:8	0xff	NVJPG: Mask NVJPG rail
7:0	0xff	NVDEC: Mask NVDEC rail

12.6.181 APBDEV_PMC_PG_MASK_6_0

Offset: 0x2e0 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	VE2: Mask VE2 rail

12.6.182 APBDEV_PMC_SECURE_SCRATCH8_0

Secure Scratch Register

Offset: 0x300 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH8

12.6.183 APBDEV_PMC_SECURE_SCRATCH9_0

Secure Scratch Register

Offset: 0x304 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH9

12.6.184 APBDEV_PMC_SECURE_SCRATCH10_0

Secure Scratch Register

Offset: 0x308 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH10

12.6.185 APBDEV_PMC_SECURE_SCRATCH11_0

Secure Scratch Register

Offset: 0x30c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH11

12.6.186 APBDEV_PMC_SECURE_SCRATCH12_0

Secure Scratch Register

Offset: 0x310 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH12

12.6.187 APBDEV_PMC_SECURE_SCRATCH13_0

Secure Scratch Register

Offset: 0x314 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH13

12.6.188 APBDEV_PMC_SECURE_SCRATCH14_0

Secure Scratch Register

Offset: 0x318 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH14

12.6.189 APBDEV_PMC_SECURE_SCRATCH15_0

Secure Scratch Register

Offset: 0x31c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH15

12.6.190 APBDEV_PMC_SECURE_SCRATCH16_0

Secure Scratch Register

Offset: 0x320 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH16

12.6.191 APBDEV_PMC_SECURE_SCRATCH17_0

Secure Scratch Register

Offset: 0x324 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH17

12.6.192 APBDEV_PMC_SECURE_SCRATCH18_0

Secure Scratch Register

Offset: 0x328 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH18

12.6.193 APBDEV_PMC_SECURE_SCRATCH19_0

Secure Scratch Register

Offset: 0x32c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH19

12.6.194 APBDEV_PMC_SECURE_SCRATCH20_0

Secure Scratch Register

Offset: 0x330 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH20

12.6.195 APBDEV_PMC_SECURE_SCRATCH21_0

Secure Scratch Register

Offset: 0x334 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH21

12.6.196 APBDEV_PMC_SECURE_SCRATCH22_0

Secure Scratch Register

Offset: 0x338 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH22

12.6.197 APBDEV_PMC_SECURE_SCRATCH23_0

Secure Scratch Register

Offset: 0x33c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH23

12.6.198 APBDEV_PMC_SECURE_SCRATCH24_0

Secure Scratch Register

Offset: 0x340 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH24

12.6.199 APBDEV_PMC_SECURE_SCRATCH25_0

Secure Scratch Register

Offset: 0x344 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH25

12.6.200 APBDEV_PMC_SECURE_SCRATCH26_0

Secure Scratch Register

Offset: 0x348 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH26

12.6.201 APBDEV_PMC_SECURE_SCRATCH27_0

Secure Scratch Register

Offset: 0x34c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH27

12.6.202 APBDEV_PMC_SECURE_SCRATCH28_0

Secure Scratch Register

Offset: 0x350 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH28

12.6.203 APBDEV_PMC_SECURE_SCRATCH29_0

Secure Scratch Register

Offset: 0x354 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH29

12.6.204 APBDEV_PMC_SECURE_SCRATCH30_0

Secure Scratch Register

Offset: 0x358 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH30

12.6.205 APBDEV_PMC_SECURE_SCRATCH31_0

Secure Scratch Register

Offset: 0x35c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH31

12.6.206 APBDEV_PMC_SECURE_SCRATCH32_0

Secure scratch register

Offset: 0x360 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH32

12.6.207 APBDEV_PMC_SECURE_SCRATCH33_0

Secure scratch register

Offset: 0x364 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH33

12.6.208 APBDEV_PMC_SECURE_SCRATCH34_0

Secure scratch register

Offset: 0x368 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH34

12.6.209 APBDEV_PMC_SECURE_SCRATCH35_0

Secure scratch register

Offset: 0x36c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH35

12.6.210 APBDEV_PMC_SECURE_SCRATCH36_0

Secure scratch register

Offset: 0x370 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH36

12.6.211 APBDEV_PMC_SECURE_SCRATCH37_0

Secure scratch register

Offset: 0x374 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH37

12.6.212 APBDEV_PMC_SECURE_SCRATCH38_0

Secure scratch register

Offset: 0x378 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH38

12.6.213 APBDEV_PMC_SECURE_SCRATCH39_0

Secure scratch register

Offset: 0x37c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH39

12.6.214 APBDEV_PMC_SECURE_SCRATCH40_0

Secure scratch register

Offset: 0x380 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH40

12.6.215 APBDEV_PMC_SECURE_SCRATCH41_0

Secure scratch register

Offset: 0x384 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH41

12.6.216 APBDEV_PMC_SECURE_SCRATCH42_0

Secure scratch register

Offset: 0x388 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH42

12.6.217 APBDEV_PMC_SECURE_SCRATCH43_0

Secure scratch register

Offset: 0x38c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH43

12.6.218 APBDEV_PMC_SECURE_SCRATCH44_0

Secure scratch register

Offset: 0x390 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH44

12.6.219 APBDEV_PMC_SECURE_SCRATCH45_0

Secure scratch register

Offset: 0x394 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH45

12.6.220 APBDEV_PMC_SECURE_SCRATCH46_0

Secure scratch register

Offset: 0x398 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH46

12.6.221 APBDEV_PMC_SECURE_SCRATCH47_0

Secure scratch register

Offset: 0x39c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH47

12.6.222 APBDEV_PMC_SECURE_SCRATCH48_0

Secure scratch register

Offset: 0x3a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH48

12.6.223 APBDEV_PMC_SECURE_SCRATCH49_0

Secure scratch register

Offset: 0x3a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH49

12.6.224 APBDEV_PMC_SECURE_SCRATCH50_0

Secure scratch register

Offset: 0x3a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH50

12.6.225 APBDEV_PMC_SECURE_SCRATCH51_0

Secure scratch register

Offset: 0x3ac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH51

12.6.226 APBDEV_PMC_SECURE_SCRATCH52_0

Secure scratch register

Offset: 0x3b0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH52

12.6.227 APBDEV_PMC_SECURE_SCRATCH53_0

Secure scratch register

Offset: 0x3b4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH53

12.6.228 APBDEV_PMC_SECURE_SCRATCH54_0

Secure scratch register

Offset: 0x3b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH54

12.6.229 APBDEV_PMC_SECURE_SCRATCH55_0

Secure scratch register

Offset: 0x3bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH55

12.6.230 APBDEV_PMC_SECURE_SCRATCH56_0

Secure scratch register

Offset: 0x3c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH56

12.6.231 APBDEV_PMC_SECURE_SCRATCH57_0

Secure scratch register

Offset: 0x3c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH57

12.6.232 APBDEV_PMC_SECURE_SCRATCH58_0

Secure scratch register

Offset: 0x3c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH58

12.6.233 APBDEV_PMC_SECURE_SCRATCH59_0

Secure scratch register

Offset: 0x3cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH59

12.6.234 APBDEV_PMC_SECURE_SCRATCH60_0

Secure scratch register

Offset: 0x3d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH60

12.6.235 APBDEV_PMC_SECURE_SCRATCH61_0

Secure scratch register

Offset: 0x3d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH61

12.6.236 APBDEV_PMC_SECURE_SCRATCH62_0

Secure scratch register

Offset: 0x3d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH62

12.6.237 APBDEV_PMC_SECURE_SCRATCH63_0

Secure scratch register

Offset: 0x3dc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH63

12.6.238 APBDEV_PMC_SECURE_SCRATCH64_0

Secure scratch register

Offset: 0x3e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH64

12.6.239 APBDEV_PMC_SECURE_SCRATCH65_0

Secure scratch register

Offset: 0x3e4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH65

12.6.240 APBDEV_PMC_SECURE_SCRATCH66_0

Secure scratch register

Offset: 0x3e8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH66

12.6.241 APBDEV_PMC_SECURE_SCRATCH67_0

Secure scratch register

Offset: 0x3ec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH67

12.6.242 APBDEV_PMC_SECURE_SCRATCH68_0

Secure scratch register

Offset: 0x3f0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH68

12.6.243 APBDEV_PMC_SECURE_SCRATCH69_0

Secure scratch register

Offset: 0x3f4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH69

12.6.244 APBDEV_PMC_SECURE_SCRATCH70_0

Secure scratch register

Offset: 0x3f8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH70

12.6.245 APBDEV_PMC_SECURE_SCRATCH71_0

Secure scratch register

Offset: 0x3fc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH71

12.6.246 APBDEV_PMC_SECURE_SCRATCH72_0

Secure scratch register

Offset: 0x400 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH72

12.6.247 APBDEV_PMC_SECURE_SCRATCH73_0

Secure scratch register

Offset: 0x404 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH73

12.6.248 APBDEV_PMC_SECURE_SCRATCH74_0

Secure scratch register

Offset: 0x408 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH74

12.6.249 APBDEV_PMC_SECURE_SCRATCH75_0

Secure scratch register

Offset: 0x40c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH75

12.6.250 APBDEV_PMC_SECURE_SCRATCH76_0

Secure scratch register

Offset: 0x410 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH76

12.6.251 APBDEV_PMC_SECURE_SCRATCH77_0

Secure scratch register

Offset: 0x414 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH77

12.6.252 APBDEV_PMC_SECURE_SCRATCH78_0

Secure scratch register

Offset: 0x418 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH78

12.6.253 APBDEV_PMC_SECURE_SCRATCH79_0

Secure scratch register

Offset: 0x41c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH79

12.6.254 APBDEV_PMC_CNTRL2_0

Wake interrupt and LP0BB Controls

Any write to CNTRL2 register with LP0_STS bit “0” will clear LP0_STS. To program any field of this register in LP0BB, set the LP0_STS bit of the write data to “1” so that LP0_STS register does not get cleared unintentionally. Note that writing 1 to LP0_STS has no impact. The LP0_STS bit cannot be set by software. It can only be cleared by software.

Offset: 0x440 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxx0x0000xxxxxxxx1)

Bit	Reset	Description
14	0x0	ALLOW_PULSE_WAKE:Allows pulse wakes in case of debounce events and/or USB events. 0 = DISABLE 1 = ENABLE
12	0x0	HOLD_CKE_LOW_EN: Defunct. Enable for the PMC to assert CKE low 0 = DISABLE 1 = ENABLE
11	0x0	SYSCLK_DATA: SYS_CLK_REQ data value. This is used when SYSCLK_ORRIDE is set to '1'
10	0x0	SYSCLK_ORRIDE: SYS_CLK_REQ data override mux control 0: data is driven by hardware 1: data is driven by SYSCLK_DATA 0 = DISABLE 1 = ENABLE
9	0x0	WAKE_DET_EN: When this bit is set, then LP0 wake events mechanism is enabled to detect wake-events The following programming sequence is needed before setting WAKE_DET_EN bit if S/W wants to reprogram PMC_WAKE*_MASK/LVL registers. PMC_WAKE*_MASK/LVL registers should only be changed when WAKE_DET_EN is cleared. Programming sequence for WAKE_DET_EN and PMC_WAKE*_MASK/LVL registers: 1. WAKE_DET_EN should be "0" or cleared.2. Program WAKE_LVL registers. Wait for some time (>2 32kHz ticks) for it to stabilize. This step can be skipped if we want to use default value of WAKE_LVL register.3. Program WAKE_MASK registers.4. Set WAKE_DET_EN 0 = DISABLE 1 = ENABLE
8:1	X	RESERVED: Reserved

Bit	Reset	Description
0	0x1	WAKE_INT_EN: When this bit is set, then LP0 wake events are used to generate a WakeInterrupt interrupt. 0 = DISABLE 1 = ENABLE

12.6.255 APBDEV_PMC_IO_DPD_OFF_MASK_0

DPD mask

Offset: 0x444 | Read/Write: R/W | Reset: 0x01f00000 (0bxxx000011111000000x00000x0000000)

Bit	Reset	Description
28	0x0	HDMI: 0 = OFF 1 = ON
27	0x0	GPIO: 0 = OFF 1 = ON
26	0x0	DEBUG_NONAO: 0 = OFF 1 = ON
25	0x0	DBG: 0 = OFF 1 = ON
24	0x1	POP_ADDR_CMD: 0 = OFF 1 = ON
23	0x1	POP_CLK: 0 = OFF 1 = ON
22	0x1	COMP: 0 = OFF 1 = ON
21	0x1	DISC_VTTGEN: 0 = OFF 1 = ON
20	0x1	POP_VTTGEN: 0 = OFF 1 = ON
19	0x0	HSIC: 0 = OFF 1 = ON
18	0x0	USB3:USB3 pads 0 = OFF 1 = ON
17	0x0	AUDIO: 0 = OFF 1 = ON
16	0x0	VI: DEFUNCT 0 = OFF 1 = ON
15	0x0	BB: DEFUNCT 0 = OFF 1 = ON
14	0x0	UART: 0 = OFF 1 = ON
12	0x0	USB_BIAS: 0 = OFF 1 = ON
11	0x0	USB2: DEFUNCT 0 = OFF 1 = ON

Bit	Reset	Description
10	0x0	USB1: 0 = OFF 1 = ON
9	0x0	USB0: 0 = OFF 1 = ON
8	0x0	DDR0_CS1_CKE1:DSC CS1 and CKE1 0 = OFF 1 = ON
6	0x0	PEX_CLK2:pex clk2 pad-DPD control 0 = OFF 1 = ON
5	0x0	PEX_CLK1:pex clk1 pad-DPD control 0 = OFF 1 = ON
4	0x0	PEX_BIAS:pex bias pad-DPD control 0 = OFF 1 = ON
3	0x0	MIPI_BIAS: 0 = OFF 1 = ON
2	0x0	DSI: This is named as DSIA in pinout specification. 0 = OFF 1 = ON
1	0x0	CSIB: 0 = OFF 1 = ON
0	0x0	CSIA: 0 = OFF 1 = ON

12.6.256 APBDEV_PMC_IO_DPD2_OFF_MASK_0

DPD off mask need second set due to additional rails

Offset: 0x448 | Read/Write: R/W | Reset: 0x1df30000 (0bxx01110111110011000000000000000)

Bit	Reset	Description
29	0x0	AUDIO_HV:
28	0x1	POP_BIAS:DPD_OFF_MASK for BG BIAS cells of POP pads 0 = OFF 1 = ON
27	0x1	DISC_BIAS:DPD_OFF_MASK for BG BIAS cells of DISC pads 0 = OFF 1 = ON
26	0x1	SYS_DDC: if DDR_DISC_HAS_BG_BIAS and DDR_DISC_HAS_BG_BIAS == 1 0 = OFF 1 = ON
25	0x0	LVDS: RESERVED (DEFUNCT) 0 = OFF 1 = ON
24	0x1	DDR1_DATA: DEFUNCT 0 = OFF 1 = ON
23	0x1	DDR1_CLK: DEFUNCT 0 = OFF 1 = ON
22	0x1	DDR1_ADDR2_CMD: DEFUNCT 0 = OFF 1 = ON

Bit	Reset	Description
21	0x1	DDR1_ADDR1_CMD: DEFUNCT 0 = OFF 1 = ON
20	0x1	DDR1_ADDR0_CMD: DEFUNCT 0 = OFF 1 = ON
19	0x0	DP: 0 = OFF 1 = ON
18	0x0	DMIC: 0 = OFF 1 = ON
17	0x1	DDR0_ADDR1_CMD: 0 = OFF 1 = ON
16	0x1	DDR0_ADDR0_CMD: 0 = OFF 1 = ON
15	0x0	SPI_HV: 0 = OFF 1 = ON
14	0x0	SPI: 0 = OFF 1 = ON
13	0x0	CSIF: 0 = OFF 1 = ON
12	0x0	CSIE: 0 = OFF 1 = ON
11	0x0	CSID: DEFUNCT 0 = OFF 1 = ON
10	0x0	CSIC: DEFUNCT 0 = OFF 1 = ON
9	0x0	DSID: DEFUNCT 0 = OFF 1 = ON
8	0x0	DSIC: DEFUNCT 0 = OFF 1 = ON
7	0x0	DSIB: 0 = OFF 1 = ON
6	0x0	HV: DEFUNCT 0 = OFF 1 = ON
5	0x0	EMMC2: 0 = OFF 1 = ON
4	0x0	CAM: 0 = OFF 1 = ON
3	0x0	EMMC: 0 = OFF 1 = ON
2	0x0	SDMMC3: 0 = OFF 1 = ON

Bit	Reset	Description
1	0x0	SDMMC1: 0 = OFF 1 = ON
0	0x0	PEX_CNTRL: DEFUNCT 0 = OFF 1 = ON

12.6.257 APBDEV_PMC_EVENT_COUNTER_0

Offset: 0x44c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20	0x0	EN:COUNT enable/disable. 0: disable, 1: enable 0 = OFF 1 = ON
19:16	0x0	SEL: Event selections. 0: LP0 to LP0BB transition event 1: LP0 to ACTIVE transition event 2: LP0BB to ACTIVE transition event 3: LP0BB to LP0 transition event 0 = OFF 1 = ON
15:0	0x0	COUNT: Event counter. Any write to this register will clear COUNT

12.6.258 APBDEV_PMC_FUSE_CONTROL_0

Offset: 0x450 | Read/Write: R/W | Reset: 0x000X0X00 (0bxxxxxxxxxxxxxxx10xxxxxx10xxxxxx00)

Bit	R/W	Reset	Description
18	RO	X	KPS18_LATCH_STATUS: reset value x
17	RW	0x1	KPS18_LATCH_CLEAR: reset value 1
16	RW	0x0	KPS18_LATCH_SET: reset value 0
10	RO	X	PS18_STATE:Status of PS18 signal.
9	RW	0x1	PS18_LATCH_CLEAR:Bit used to clear the latch driving PS18.
8	RW	0x0	PS18_LATCH_SET:Bit used to set the latch driving PS18.
1	RW	0x0	DISABLE_REDIRECTION: sticky disable redirection bit, reset at cold and warm reset 0 = OFF 1 = ON
0	RW	0x0	ENABLE_REDIRECTION: enable redirection bit - only reset at cold reset. This bit can be modified at will but would typically be set after each cold reset, possibly based on a bit in the BCT or by some other software based configuration mechanism 0 = OFF 1 = ON

12.6.259 APBDEV_PMC_SCRATCH1_ECO_0

Scratch register

Offset: 0x454 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	ECO1:Reserved

12.6.260 APBDEV_PMC_IO_DPD3_REQ_0

DPD Request 3 Register

Note: *Put DDR reset pad in DPD prior to entering LP0 to make sure the reset does not become 'z' or 'x' while entering/exiting LP0.*

Offset: 0x45c | Read/Write: R/W | Reset: 0x0fff0000 (0b00xx11111111111000000000000000)

Bit	Reset	Description
31:30	0x0	CODE: code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
27	ON	DDR_BR11_CMD: 0 = OFF 1 = ON
26	ON	DDR_BR10_CMD: 0 = OFF 1 = ON
25	ON	DDR_BR9_CMD: 0 = OFF 1 = ON
24	ON	DDR_BR8_CMD: 0 = OFF 1 = ON
23	ON	DDR_BR7_CMD: 0 = OFF 1 = ON
22	ON	DDR_BR6_CMD: 0 = OFF 1 = ON
21	ON	DDR_BR5_CMD: 0 = OFF 1 = ON
20	ON	DDR_BR4_CMD: 0 = OFF 1 = ON
19	ON	DDR_BR3_CMD: 0 = OFF 1 = ON
18	ON	DDR_BR2_CMD: 0 = OFF 1 = ON
17	ON	DDR_BR1_CMD: 0 = OFF 1 = ON
16	ON	DDR_BR0_CMD: 0 = OFF 1 = ON
15	0x0	DDR_DDLL: 0 = OFF 1 = ON
14	0x0	DDR_CDBBUF1: 0 = OFF 1 = ON
13	0x0	DDR_CDBBUF0: 0 = OFF 1 = ON
12	0x0	DDR_COMP: 0 = OFF 1 = ON

Bit	Reset	Description
11	0x0	DDR_BR11: 0 = OFF 1 = ON
10	0x0	DDR_BR10: 0 = OFF 1 = ON
9	0x0	DDR_BR9: 0 = OFF 1 = ON
8	0x0	DDR_BR8: 0 = OFF 1 = ON
7	0x0	DDR_BR7: 0 = OFF 1 = ON
6	0x0	DDR_BR6: 0 = OFF 1 = ON
5	0x0	DDR_BR5: 0 = OFF 1 = ON
4	0x0	DDR_BR4: 0 = OFF 1 = ON
3	0x0	DDR_BR3: 0 = OFF 1 = ON
2	0x0	DDR_BR2: 0 = OFF 1 = ON
1	0x0	DDR_BR1: 0 = OFF 1 = ON
0	0x0	DDR_BR0: 0 = OFF 1 = ON

12.6.261 APBDEV_PMC_IO_DPD3_STATUS_0

DPD Status 3 Register

Offset: 0x460 | Read/Write: R/W | Reset: 0x0fff0000 (0bxxxx11111111111100000000000000)

Bit	Reset	Description
27	ON	DDR_BR11_CMD: 0 = OFF 1 = ON
26	ON	DDR_BR10_CMD: 0 = OFF 1 = ON
25	ON	DDR_BR9_CMD: 0 = OFF 1 = ON
24	ON	DDR_BR8_CMD: 0 = OFF 1 = ON
23	ON	DDR_BR7_CMD: 0 = OFF 1 = ON
22	ON	DDR_BR6_CMD: 0 = OFF 1 = ON

Bit	Reset	Description
21	ON	DDR_BR5_CMD: 0 = OFF 1 = ON
20	ON	DDR_BR4_CMD: 0 = OFF 1 = ON
19	ON	DDR_BR3_CMD: 0 = OFF 1 = ON
18	ON	DDR_BR2_CMD: 0 = OFF 1 = ON
17	ON	DDR_BR1_CMD: 0 = OFF 1 = ON
16	ON	DDR_BR0_CMD: 0 = OFF 1 = ON
15	OFF	DDR_DDLL: 0 = OFF 1 = ON
14	OFF	DDR_CDBBUF1: 0 = OFF 1 = ON
13	OFF	DDR_CDBBUF0: 0 = OFF 1 = ON
12	OFF	DDR_COMP: 0 = OFF 1 = ON
11	OFF	DDR_BR11: 0 = OFF 1 = ON
10	OFF	DDR_BR10: 0 = OFF 1 = ON
9	OFF	DDR_BR9: 0 = OFF 1 = ON
8	OFF	DDR_BR8: 0 = OFF 1 = ON
7	OFF	DDR_BR7: 0 = OFF 1 = ON
6	OFF	DDR_BR6: 0 = OFF 1 = ON
5	OFF	DDR_BR5: 0 = OFF 1 = ON
4	OFF	DDR_BR4: 0 = OFF 1 = ON
3	OFF	DDR_BR3: 0 = OFF 1 = ON
2	OFF	DDR_BR2: 0 = OFF 1 = ON

Bit	Reset	Description
1	OFF	DDR_BR1: 0 = OFF 1 = ON
0	OFF	DDR_BR0: 0 = OFF 1 = ON

12.6.262 APBDEV_PMC_IO_DPD4_REQ_0

DPD request 4 register

Offset: 0x464 | Read/Write: R/W | Reset: 0x00000000 (0b00xx000000000000xxx0000000000000)

Bit	Reset	Description
31:30	0x0	CODE: code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
27	0x0	DDR_BR11_VTTGEN: 0 = OFF 1 = ON
26	0x0	DDR_BR10_VTTGEN: 0 = OFF 1 = ON
25	0x0	DDR_BR9_VTTGEN: 0 = OFF 1 = ON
24	0x0	DDR_BR8_VTTGEN: 0 = OFF 1 = ON
23	0x0	DDR_BR7_VTTGEN: 0 = OFF 1 = ON
22	0x0	DDR_BR6_VTTGEN: 0 = OFF 1 = ON
21	0x0	DDR_BR5_VTTGEN: 0 = OFF 1 = ON
20	0x0	DDR_BR4_VTTGEN: 0 = OFF 1 = ON
19	0x0	DDR_BR3_VTTGEN: 0 = OFF 1 = ON
18	0x0	DDR_BR2_VTTGEN: 0 = OFF 1 = ON
17	0x0	DDR_BR1_VTTGEN: 0 = OFF 1 = ON
16	0x0	DDR_BR0_VTTGEN: 0 = OFF 1 = ON
12	0x0	DDR_COMP_BG: 0 = OFF 1 = ON
11	0x0	DDR_BR11_BG: 0 = OFF 1 = ON
10	0x0	DDR_BR10_BG: 0 = OFF 1 = ON

Bit	Reset	Description
9	0x0	DDR_BR9_BG: 0 = OFF 1 = ON
8	0x0	DDR_BR8_BG: 0 = OFF 1 = ON
7	0x0	DDR_BR7_BG: 0 = OFF 1 = ON
6	0x0	DDR_BR6_BG: 0 = OFF 1 = ON
5	0x0	DDR_BR5_BG: 0 = OFF 1 = ON
4	0x0	DDR_BR4_BG: 0 = OFF 1 = ON
3	0x0	DDR_BR3_BG: 0 = OFF 1 = ON
2	0x0	DDR_BR2_BG: 0 = OFF 1 = ON
1	0x0	DDR_BR1_BG: 0 = OFF 1 = ON
0	0x0	DDR_BR0_BG: 0 = OFF 1 = ON

12.6.263 APBDEV_PMC_IO_DPD4_STATUS_0

DPD status 4 register

Offset: 0x468 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxx0000000000000)

Bit	Reset	Description
27	0x0	DDR_BR11_VTTGEN: 0 = OFF 1 = ON
26	0x0	DDR_BR10_VTTGEN: 0 = OFF 1 = ON
25	0x0	DDR_BR9_VTTGEN: 0 = OFF 1 = ON
24	0x0	DDR_BR8_VTTGEN: 0 = OFF 1 = ON
23	0x0	DDR_BR7_VTTGEN: 0 = OFF 1 = ON
22	0x0	DDR_BR6_VTTGEN: 0 = OFF 1 = ON
21	0x0	DDR_BR5_VTTGEN: 0 = OFF 1 = ON
20	0x0	DDR_BR4_VTTGEN: 0 = OFF 1 = ON

Bit	Reset	Description
19	0x0	DDR_BR3_VTTGEN: 0 = OFF 1 = ON
18	0x0	DDR_BR2_VTTGEN: 0 = OFF 1 = ON
17	0x0	DDR_BR1_VTTGEN: 0 = OFF 1 = ON
16	0x0	DDR_BR0_VTTGEN: 0 = OFF 1 = ON
12	0x0	DDR_COMP_BG: 0 = OFF 1 = ON
11	0x0	DDR_BR11_BG: 0 = OFF 1 = ON
10	0x0	DDR_BR10_BG: 0 = OFF 1 = ON
9	0x0	DDR_BR9_BG: 0 = OFF 1 = ON
8	0x0	DDR_BR8_BG: 0 = OFF 1 = ON
7	0x0	DDR_BR7_BG: 0 = OFF 1 = ON
6	0x0	DDR_BR6_BG: 0 = OFF 1 = ON
5	0x0	DDR_BR5_BG: 0 = OFF 1 = ON
4	0x0	DDR_BR4_BG: 0 = OFF 1 = ON
3	0x0	DDR_BR3_BG: 0 = OFF 1 = ON
2	0x0	DDR_BR2_BG: 0 = OFF 1 = ON
1	0x0	DDR_BR1_BG: 0 = OFF 1 = ON
0	0x0	DDR_BR0_BG: 0 = OFF 1 = ON

12.6.264 APBDEV_PMC_DIRECT_THERMTRIP_CFG_0

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010000)

Bit	Reset	Description
5	0x0	THERMTRIP_LOCK: When set to 1, locks THERMTRIP_EN and AOTAG THERMTRIP threshold. This lock bit is expected to write-lock all register bits that can disable or sabotage direct shutdown. Write-once Updated by secure software after setting all direct shutdown thermtrip related registers. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x1	THERMTRIP_EN: When this bit is set, then PMC will assert SHUTDOWN sideband to PMIC on a thermal sensor reset event. The system will then be shutdown directly by PMIC instead of causing a shutdown through Boot ROM. This configuration bit will be cleared only by an external reset. The thermal sensor reset request can come from SOC_therm or from the internal AOTAG in PMC. SOC_therm will not be able to send a reset request while in LP0 or LP0BB. The internal AOTAG will be capable of sending a thermtrip alert in LP0 or LP0BB. This field overrides PMC_SENSOR_CTRL_ENABLE_RST. 0 = DISABLE 1 = ENABLE
3:0	0x0	SHDN_ASSERT_PULSE_WIDTH: SHUTDOWN pin, once asserted by PMC, remains asserted for the programmed number of 32K clock cycles. This field is cleared by external reset only

12.6.265 APBDEV_PMC_TSOSC_DELAY_0

Offset: 0x478 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxx000001111111)

Bit	Reset	Description
11:0	0x7f	TSOSC_PWRGOOD_DLY: Delay before re-enabling CCPLEX TSOSCs while exiting from CC4. Timer value is TSOSC_PWRGOOD_DLY number of osc clock cycles

12.6.266 APBDEV_PMC_SET_SW_CLAMP_0

Offset: 0x47c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CRAIL: If this bit is set CRAIL clamp is set. If this bit is cleared, PMC drives the real clamp value out. CLAMP_STATUS[CRAIL] also gets set, if this bit is set.

12.6.267 APBDEV_PMC_DEBUG_AUTHENTICATION_0

Offset: 0x480 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5	0x1	DBGEN
4	0x1	NIDEN
3	0x1	SPIDEN
2	0x1	SPNIDEN
1	0x1	DEVICEEN
0	0x1	JTAG_ENABLE

12.6.268 APBDEV_PMC_AOTAG_CFG_0

PMC AOTAG related registers

Offset: 0x484 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0000011)

Bit	R/W	Reset	Description
6	RW	0x0	AOTAG_THERMTRIP_EN: when set enables thermtrip from AOTAG, if direct thermtrip enable is set it will cause shutdown assertion or it causes reset assertion (legacy mode)
5	RW	0x0	TAG_EN: Enable function
4	RW	0x0	ENABLE_SW_DEBUG_MODE: Setting to '1' enables software debug mode. With this bit =1, writing 1 to SW_TRIGGER triggers FSMUSE_EXT_TRIGGER = 1 overrides this. USE_EXT_TRIGGER = 0 && ENABLE_SW_DEBUG_MODE = 0 will put the sensing FSM in periodic mode
3	RW	0x0	USE_EXT_TRIGGER: Internal timer vs OR of other system events like paging.
2	WO	0x0	SW_TRIGGER: Internal timer vs software trigger for debug. When set to 1, this triggers the start of the temperature sensing.
1	RW	0x1	RESERVED: reserved for future
0	RW	0x1	DISABLE_CLK_GATING: Force enable all clock gating

12.6.269 APBDEV_PMC_AOTAG_THRESH1_CFG_0

Offset: 0x488 | Read/Write: R/W | Reset: 0x007f80fa (0bxx00000011111111000000011111010)

Bit	Reset	Description
29:15	0xff	HOT_SET: Hot indicator set threshold
14:0	0xfa	HOT_RESET: Hot indicator reset threshold

12.6.270 APBDEV_PMC_AOTAG_THRESH2_CFG_0

Offset: 0x48c | Read/Write: R/W | Reset: 0x00000005 (0bxx00000000000000000000000000101)

Bit	Reset	Description
29:15	0x0	COLD_SET: Cold indicator set threshold
14:0	0x5	COLD_RESET: Cold indicator reset threshold

12.6.271 APBDEV_PMC_AOTAG_THRESH3_CFG_0

Offset: 0x490 | Read/Write: R/W | Reset: 0x000000fa (0bxxxxxxxxxxxxxxxxxx000000011111010)

Bit	Reset	Description
14:0	0xfa	THERMTRIP_SET: Thermtrip indicator set threshold

12.6.272 APBDEV_PMC_AOTAG_STATUS_0

Offset: 0x494 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	THERMTRIP: Write to PMC_TAG_STATUS resets this field to 0. The thermtrip reset status is also temporarily captured in PMC_RST_STATUS register in case thermtrip solution uses Boot ROM to reset the chip. 0 = NOTSET 1 = SET
1	X	TEMP_LOW: Write to PMC_TAG_STATUS resets this field to 0 0 = NOTSET 1 = SET
0	X	TEMP_HIGH: Write to PMC_TAG_STATUS resets this field to 0 0 = NOTSET 1 = SET

12.6.273 APBDEV_PMC_AOTAG_SECURITY_0

Offset: 0x498 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	WR_LOCK_ALL: Write lock all PMC_AOTAG registers. Cleared by power-on reset.
2	0x0	WR_LOCK_HOT_THRESHOLD: Write lock PMC_AOTAG_THRESH1_CFG. Cleared by power-on reset.
1	0x0	WR_LOCK_COLD_THRESHOLD: Write lock PMC_AOTAG_THRESH2_CFG. Cleared by power-on reset.
0	0x0	WR_LOCK_THERMTRIP: Write lock PMC_AOTAG_THRESH3_CFG. Cleared by power-on reset. Also write locks all other thermtrip related registers in the system.

12.6.274 APBDEV_PMC_TSENSOR_CONFIG0_0

Offset: 0x49c | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000001)

Bit	Reset	Description
31:28	0x0	RESERVED_1:Reserved
27:8	0x0	TALL: M count. Time in tsensor_clk for full capture cycle. To reduce power, its value should be at least 100 times larger than TSAMPLE.

Bit	Reset	Description
7:6	0x0	RESERVED_2:Reserved
5	0x0	STATUS_CLR: Clears MIN/MAX statistics. Clears MIN/MAX statistics. Write 1 to clear. Readback always is 0.
4	0x0	TCALC_OVERFLOW: add operation overflow. sticky bit, Indicates that overflow happened during temperature conversion. Never should happen. Used to verify that coefficients don't cause overflow. Cleared by writing 1 to this bit.
3	0x0	OVERFLOW: 14 bit overflow. sticky bit. Set if capture counter overflows 14 bits used for temperature translation. Cleared by writing 1 to this bit.
2	0x0	CPTR_OVERFLOW: capture overflow. sticky bit. Set if capture counter overflows during the capture. Cleared by writing 1 to this bit
1	0x0	RO_SEL: ring oscillator select. selects between Temperature and Voltage output for the capture, 0 = TS_OSC_OUT, 1 = VS_OSC_OUT. This bit should always be set to 0 in the PMC_AOTAG instance of this register
0	0x1	STOP: sensor stop. Disables the sensor from taking any reading, stops the ring oscillators. set this bit to 1 to stop the sensor from taking measurements, set to 0 to start taking measurements. Should be set by SW if VDD_RTC ever goes below the TSOSC Vddmin. In PMC_AOTAG this bit is redundant to the general AOTAG enable bit.

12.6.275 APBDEV_PMC_TSENSOR_CONFIG1_0

Offset: 0x4a0 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: temperature enable. Enables temperature translation if set. If cleared, corresponding temperature is forced to read 0C. This bit should be set to 1 before enabling PMC_AOTAG in normal operation.
29:24	0x0	TEN_COUNT: time between en and capture
20:15	0x0	TIDDQ_EN: time between iddq and en
9:0	0x0	TSAMPLE: N count

12.6.276 APBDEV_PMC_TSENSOR_CONFIG2_0

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

12.6.277 APBDEV_PMC_TSENSOR_STATUS0_0

Offset: 0x4a8 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: valid capture available
15:0	X	CAPTURE: last valid raw capture

12.6.278 APBDEV_PMC_TSENSOR_STATUS1_0

Offset: 0x4ac | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: valid capture available
15:0	X	TEMP: last valid translated temp

12.6.279 APBDEV_PMC_TSENSOR_STATUS2_0

Offset: 0x4b0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: max temp registered since last clear, reset to lowest possible temp
15:0	X	TEMP_MIN: min temp registered since last clear, reset to highest possible temp

12.6.280 APBDEV_PMC_TSENSOR_PDIV_0

Offset: 0x4b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xxxxxxxxxxxx)

Bit	Reset	Description
15:12	0x0	PDIV:TSOSC post divider setting

12.6.281 APBDEV_PMC_AOTAG_INTR_EN_0

Offset: 0x4b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	HOT: Enable hot interrupt, writing '1' sets it and reading gives the current status of the bit
0	0x0	COLD: Enable cold interrupt, writing '1' sets it and reading gives the current status of the bit

12.6.282 APBDEV_PMC_AOTAG_INTR_DIS_0

Offset: 0x4bc | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	HOT: Enable hot interrupt, writing '1' clears the enable bit and reading gives '0'
0	0x0	COLD: Enable cold interrupt, writing '1' clears the enable bit and reading gives '0'

12.6.283 APBDEV_PMC_UTMIP_PAD_CFG0_0

IO Pad config for port 0

Set of static config values for USB IO pads under DPD mode

These were config values captured at a time prior to entering DPD.

These values are read only. Registers must take into account DFT.

Offset: 0x4c0 | Read/Write: R/W | Reset: 0x28XXXXX (0bxx10100001xxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
29	RW	0x1	RPU_SWITCH_LOW_P0
28	RW	0x0	RPU_HIGH_FIXED_P0: IO pad RPU_HIGH_FIXED for UTMIP P0
27	RW	0x1	RPU_AUTO_SWITCH_EN_P0: IO pad RPU_AUTO_SWITCH_EN for UTMIP P0
26:22	RW	0x1	RPD_CTRL_P0: IO pad RPD_CTRL for UTMIP P0
21	RO	X	RPU_STATUS_HIGH_P0: IO pad RPU_STATUS_HIGH for UTMIP P0
20	RO	X	LO_SPD_P0: IO pad LO_SPD for UTMIP P0
19:16	RO	X	SPARE_P0: IO pad SPARE1.0 for UTMIP P0
15:12	RO	X	FS_FSLEW_P0: IO pad FS_SLEW1..0 for UTMIP P0
11:8	RO	X	FS_RSLEW_P0: IO pad FS_SLEW0..1 for UTMIP P0
7:4	RO	X	LS_FSLEW_P0: IO pad LS_FSLEW1..0 for UTMIP P0
3:0	RO	X	LS_RSLEW_P0: IO pad LS_RSLEW0..0 for UTMIP P0

12.6.284 APBDEV_PMC_UTMIP_PAD_CFG1_0

IO Pad config for port 1

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x28XXXXXX (0bxx10100001xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
29	RW	0x1	RPU_SWITCH_LOW_P1
28	RW	0x0	RPU_HIGH_FIXED_P1: IO pad RPU_HIGH_FIXED for UTMIP P1
27	RW	0x1	RPU_AUTO_SWITCH_EN_P1: IO pad RPU_AUTO_SWITCH_EN for UTMIP P1
26:22	RW	0x1	RPD_CTRL_P1: IO pad RPD_CTRL for UTMIP P1
21	RO	X	RPU_STATUS_HIGH_P1: IO pad RPU_STATUS_HIGH for UTMIP P1
20	RO	X	LO_SPD_P1: IO pad LO_SPD for UTMIP P1
19:16	RO	X	SPARE_P1: IO pad SPARE1..0 for UTMIP P1
15:12	RO	X	FS_FSLEW_P1: IO pad FS_SLEW1..0 for UTMIP P1
11:8	RO	X	FS_RSLEW_P1: IO pad FS_SLEW0..1 for UTMIP P1
7:4	RO	X	LS_FSLEW_P1: IO pad LS_FSLEW1..0 for UTMIP P1
3:0	RO	X	LS_RSLEW_P1: IO pad LS_RSLEW0..1 for UTMIP P1

12.6.285 APBDEV_PMC_UTMIP_PAD_CFG2_0

IO Pad config for port 2

Offset: 0x4c8 | Read/Write: R/W | Reset: 0x28XXXXXX (0bxx10100001xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
29	RW	0x1	RPU_SWITCH_LOW_P2
28	RW	0x0	RPU_HIGH_FIXED_P2: IO pad RPU_HIGH_FIXED for UTMIP P2
27	RW	0x1	RPU_AUTO_SWITCH_EN_P2: IO pad RPU_AUTO_SWITCH_EN for UTMIP P2
26:22	RW	0x1	RPD_CTRL_P2: IO pad RPD_CTRL for UTMIP P2
21	RO	X	RPU_STATUS_HIGH_P2: IO pad RPU_STATUS_HIGH for UTMIP P2
20	RO	X	LO_SPD_P2: IO pad LO_SPD for UTMIP P2
19:16	RO	X	SPARE_P2: IO pad SPARE1..0 for UTMIP P2
15:12	RO	X	FS_FSLEW_P2: IO pad FS_SLEW1..0 for UTMIP P2
11:8	RO	X	FS_RSLEW_P2: IO pad FS_SLEW0..1 for UTMIP P2
7:4	RO	X	LS_FSLEW_P2: IO pad LS_FSLEW1..0 for UTMIP P2
3:0	RO	X	LS_RSLEW_P2: IO pad LS_RSLEW0..1 for UTMIP P2

12.6.286 APBDEV_PMC_UTMIP_PAD_CFG3_0

IO Pad config for port 3

Offset: 0x4cc | Read/Write: R/W | Reset: 0x28XXXXXX (0bxx10100001xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
29	RW	0x1	RPU_SWITCH_LOW_P3
28	RW	0x0	RPU_HIGH_FIXED_P3: IO pad RPU_HIGH_FIXED for UTMIP P3
27	RW	0x1	RPU_AUTO_SWITCH_EN_P3: IO pad RPU_AUTO_SWITCH_EN for UTMIP P3
26:22	RW	0x1	RPD_CTRL_P3: IO pad RPD_CTRL for UTMIP P3
21	RO	X	RPU_STATUS_HIGH_P3: IO pad RPU_STATUS_HIGH for UTMIP P3
20	RO	X	LO_SPD_P3: IO pad LO_SPD for UTMIP P3
19:16	RO	X	SPARE_P3: IO pad SPARE1..0 for UTMIP P3
15:12	RO	X	FS_FSLEW_P3: IO pad FS_SLEW1..0 for UTMIP P3
11:8	RO	X	FS_RSLEW_P3: IO pad FS_SLEW0..1 for UTMIP P3

Bit	R/W	Reset	Description
7:4	RO	X	LS_FSLEW_P3: IO pad LS_FSLEW1..0 for UTMIP P3
3:0	RO	X	LS_RSLEW_P3: IO pad LS_RSLEW0..1 for UTMIP P3

12.6.287 APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0

Sleep Walk Sequence Enables

Offset: 0x4d0 | Read/Write: R/W | Reset: 0x000000c0 (0bxxxxxxxxxxxxx00000000000011000000)

Bit	Reset	Description
19	0x0	UTMIP_RPD_CTRL_USE_PMC_P3: Use PMC Saved Pad config on UTMIP P3
18	0x0	UTMIP_RPD_CTRL_USE_PMC_P2: Use PMC Saved Pad config on UTMIP P2
17	0x0	UTMIP_RPD_CTRL_USE_PMC_P1: Use PMC Saved Pad config on UTMIP P1
16	0x0	UTMIP_RPD_CTRL_USE_PMC_P0: Use PMC Saved Pad config on UTMIP P0
15:12	0x0	RESERVED_P3:reserved for future
11	0x0	UTMIP_RPU_SWITC_LOW_USE_PMC_P3: Use PMC Saved RPU_SWITC_LOW on UTMIP P3
10	0x0	UTMIP_RPU_SWITC_LOW_USE_PMC_P2: Use PMC Saved RPU_SWITC_LOW on UTMIP P2
9	0x0	UTMIP_RPU_SWITC_LOW_USE_PMC_P1: Use PMC Saved RPU_SWITC_LOW on UTMIP P1
8	0x0	UTMIP_RPU_SWITC_LOW_USE_PMC_P0: Use PMC Saved RPU_SWITC_LOW on UTMIP P0
7:4	0xc	UTMIP_WAKE_VAL_P3: Line Value Wake Up Condition on UTMIP P0 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
3	0x0	UTMIP_TCTRL_USE_PMC_P3: Use PMC Saved TCTRL on UTMIP P0
2	0x0	UTMIP_PCTRL_USE_PMC_P3: Use PMC Saved PCTRL on UTMIP P0
1	0x0	UTMIP_FSLS_USE_PMC_P3: Use PMC Saved Pad config on UTMIP P0
0	0x0	UTMIP_MASTER_ENABLE_P3: Enable use of master pins on UTMIP P0

12.6.288 APBDEV_PMC_CC4_HVC_CONTROL_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MASK_CPUPWRREQ: If this bit is set, cpu pwr req will not asserted in CC3 0 = DISABLE 1 = ENABLE

12.6.289 APBDEV_PMC_WAKE_DEBOUNCE_EN_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000xxxxxxxx00000000)

Bit	Reset	Description
23:16	0x0	DEBOUNCE_DELAY: Specifies the common de-bounce delay for wake events as (DEBOUNCE_DELAY + 1) ms. It cant be changed when debouncing is enabled
7:0	0x0	WAKE_DEBOUNCE_EN: Enable bits for debounce function for 8 GPIOs. NOTE: For debounced wake pulses to be retained in status registers, SW needs to set ALLOW_PULSE_WAKE bit in CNTRL2 register

12.6.290 APBDEV_PMC_RAMDUMP_CTL_STATUS_0

This register is used for RAM-dump feature.

These bits should be reset only on POR reset, software reset, thermtrip reset. These should survive watchdog reset and LP0 reset

Offset: 0x4dc | Read/Write: R/W | Reset: 0x000000fc (0bxxxxxxxxxxxxxxxxxxxxxx00011111100)

Bit	Reset	Description
10	0x0	RAMDUMP_STATUS_6
9	0x0	RAMDUMP_STATUS_5
8	0x0	RAMDUMP_STATUS_4
7	0x1	RAMDUMP_STATUS_3
6	0x1	RAMDUMP_STATUS_2
5	0x1	RAMDUMP_STATUS_1
4	0x1	MC_HOTRESET_COMPLETE
3	0x1	WCAM_FLUSHED
2	0x1	EMC_FLUSHED
1	0x0	L2RSTDISABLE
0	0x0	RAMDUMP_EN

12.6.291 APBDEV_PMC_UTMIP_SLEEPWALK_P3_0

Signaling Sequence for UTMIP Port 1 Wakeup

Offset: 0x4e0 | Read/Write: R/W | Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line

Bit	Reset	Description
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

12.6.292 APBDEV_PMC_DDR_CNTRL_0

Offset: 0x4e4 | Read/Write: R/W | Reset: 0x0007ff9f (0bxxxxxxxxxxxxxx111111111110011111)

Bit	Reset	Description
18	0x1	CMD_HOLD_LOW_BR11:cmd_hold_low for DDR brick11
17	0x1	CMD_HOLD_LOW_BR10:cmd_hold_low for DDR brick10
16	0x1	CMD_HOLD_LOW_BR9:cmd_hold_low for DDR brick9
15	0x1	CMD_HOLD_LOW_BR8:cmd_hold_low for DDR brick8
14	0x1	CMD_HOLD_LOW_BR7:cmd_hold_low for DDR brick7
13	0x1	CMD_HOLD_LOW_BR6:cmd_hold_low for DDR brick6
12	0x1	CMD_HOLD_LOW_BR5:cmd_hold_low for DDR brick5
11	0x1	CMD_HOLD_LOW_BR4:cmd_hold_low for DDR brick4
10	0x1	CMD_HOLD_LOW_BR3:cmd_hold_low for DDR brick3
9	0x1	CMD_HOLD_LOW_BR2:cmd_hold_low for DDR brick2
8	0x1	CMD_HOLD_LOW_BR1:cmd_hold_low for DDR brick1
7	0x1	CMD_HOLD_LOW_BR0:cmd_hold_low for DDR brick0
6:5	0x0	CMD_SEL_MODE
4	0x1	VTT_VCLAMP_E_REG
3	0x1	VTT_VDA_E_REG
2	0x1	VTT_VAUXP_E_REG
1:0	0x3	DDR_DPD_IO: common DPD I/O for DDR pads, not needed, potential can tristate

12.6.293 APBDEV_PMC_SEC_DISABLE4_0

Offset: 0x5b0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ55: disable read from secure register 55 0 = OFF 1 = ON
30	0x0	WRITE55: disable write to secure register 55 0 = OFF 1 = ON
29	0x0	READ54: disable read from secure register 54 0 = OFF 1 = ON
28	0x0	WRITE54: disable write to secure register 54 0 = OFF 1 = ON
27	0x0	READ53: disable read from secure register 53 0 = OFF 1 = ON

Bit	Reset	Description
26	0x0	WRITE53: disable write to secure register 53 0 = OFF 1 = ON
25	0x0	READ52: disable read from secure register 52 0 = OFF 1 = ON
24	0x0	WRITE52: disable write to secure register 52 0 = OFF 1 = ON
23	0x0	READ51: disable read from secure register 51 0 = OFF 1 = ON
22	0x0	WRITE51: disable write to secure register 51 0 = OFF 1 = ON
21	0x0	READ50: disable read from secure register 50 0 = OFF 1 = ON
20	0x0	WRITE50: disable write to secure register 50 0 = OFF 1 = ON
19	0x0	READ49: disable read from secure register 49 0 = OFF 1 = ON
18	0x0	WRITE49: disable write to secure register 49 0 = OFF 1 = ON
17	0x0	READ48: disable read from secure register 48 0 = OFF 1 = ON
16	0x0	WRITE48: disable write to secure register 48 0 = OFF 1 = ON
15	0x0	READ47: disable read from secure register 47 0 = OFF 1 = ON
14	0x0	WRITE47: disable write to secure register 47 0 = OFF 1 = ON
13	0x0	READ46: disable read from secure register 46 0 = OFF 1 = ON
12	0x0	WRITE46: disable write to secure register 46 0 = OFF 1 = ON
11	0x0	READ45: disable read from secure register 45 0 = OFF 1 = ON
10	0x0	WRITE45: disable write to secure register 45 0 = OFF 1 = ON
9	0x0	READ44: disable read from secure register 44 0 = OFF 1 = ON
8	0x0	WRITE44: disable write to secure register 44 0 = OFF 1 = ON
7	0x0	READ43: disable read from secure register 43 0 = OFF 1 = ON

Bit	Reset	Description
6	0x0	WRITE43: disable write to secure register 43 0 = OFF 1 = ON
5	0x0	READ42: disable read from secure register 42 0 = OFF 1 = ON
4	0x0	WRITE42: disable write to secure register 42 0 = OFF 1 = ON
3	0x0	READ41: disable read from secure register 41 0 = OFF 1 = ON
2	0x0	WRITE41: disable write to secure register 41 0 = OFF 1 = ON
1	0x0	READ40: disable read from secure register 40 0 = OFF, 1 = ON 0 = OFF 1 = ON
0	0x0	WRITE40: disable write to secure register 40 0 = OFF, 1 = ON 0 = OFF 1 = ON

12.6.294 APBDEV_PMC_SEC_DISABLE5_0

Offset: 0x5b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ71: disable read from secure register 71 0 = OFF 1 = ON
30	0x0	WRITE71: disable write to secure register 71 0 = OFF 1 = ON
29	0x0	READ70: disable read from secure register 70 0 = OFF 1 = ON
28	0x0	WRITE70: disable write to secure register 70 0 = OFF 1 = ON
27	0x0	READ69: disable read from secure register 69 0 = OFF 1 = ON
26	0x0	WRITE69: disable write to secure register 69 0 = OFF 1 = ON
25	0x0	READ68: disable read from secure register 68 0 = OFF 1 = ON
24	0x0	WRITE68: disable write to secure register 68 0 = OFF 1 = ON
23	0x0	READ67: disable read from secure register 67 0 = OFF 1 = ON
22	0x0	WRITE67: disable write to secure register 67 0 = OFF 1 = ON
21	0x0	READ66: disable read from secure register 66 0 = OFF 1 = ON

Bit	Reset	Description
20	0x0	WRITE66: disable write to secure register 66 0 = OFF 1 = ON
19	0x0	READ65: disable read from secure register 65 0 = OFF 1 = ON
18	0x0	WRITE65: disable write to secure register 65 0 = OFF 1 = ON
17	0x0	READ64: disable read from secure register 64 0 = OFF 1 = ON
16	0x0	WRITE64: disable write to secure register 64 0 = OFF 1 = ON
15	0x0	READ63: disable read from secure register 63 0 = OFF 1 = ON
14	0x0	WRITE63: disable write to secure register 63 0 = OFF 1 = ON
13	0x0	READ62: disable read from secure register 62 0 = OFF 1 = ON
12	0x0	WRITE62: disable write to secure register 62 0 = OFF 1 = ON
11	0x0	READ61: disable read from secure register 61 0 = OFF 1 = ON
10	0x0	WRITE61: disable write to secure register 61 0 = OFF 1 = ON
9	0x0	READ60: disable read from secure register 60 0 = OFF 1 = ON
8	0x0	WRITE60: disable write to secure register 60 0 = OFF 1 = ON
7	0x0	READ59: disable read from secure register 59 0 = OFF 1 = ON
6	0x0	WRITE59: disable write to secure register 59 0 = OFF 1 = ON
5	0x0	READ58: disable read from secure register 58 0 = OFF 1 = ON
4	0x0	WRITE58: disable write to secure register 58 0 = OFF 1 = ON
3	0x0	READ57: disable read from secure register 57 0 = OFF 1 = ON
2	0x0	WRITE57: disable write to secure register 57 0 = OFF 1 = ON
1	0x0	READ56: disable read from secure register 56 0 = OFF, 1 = ON 0 = OFF 1 = ON

Bit	Reset	Description
0	0x0	WRITE56: disable write to secure register 56 0 = OFF, 1 = ON 0 = OFF 1 = ON

12.6.295 APBDEV_PMC_SEC_DISABLE6_0

Offset: 0x5b8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ87: disable read from secure register 87 0 = OFF 1 = ON
30	0x0	WRITE87: disable write to secure register 87 0 = OFF 1 = ON
29	0x0	READ86: disable read from secure register 86 0 = OFF 1 = ON
28	0x0	WRITE86: disable write to secure register 86 0 = OFF 1 = ON
27	0x0	READ85: disable read from secure register 85 0 = OFF 1 = ON
26	0x0	WRITE85: disable write to secure register 85 0 = OFF 1 = ON
25	0x0	READ84: disable read from secure register 84 0 = OFF 1 = ON
24	0x0	WRITE84: disable write to secure register 84 0 = OFF 1 = ON
23	0x0	READ83: disable read from secure register 83 0 = OFF 1 = ON
22	0x0	WRITE83: disable write to secure register 83 0 = OFF 1 = ON
21	0x0	READ82: disable read from secure register 82 0 = OFF 1 = ON
20	0x0	WRITE82: disable write to secure register 82 0 = OFF 1 = ON
19	0x0	READ81: disable read from secure register 81 0 = OFF 1 = ON
18	0x0	WRITE81: disable write to secure register 81 0 = OFF 1 = ON
17	0x0	READ80: disable read from secure register 80 0 = OFF 1 = ON
16	0x0	WRITE80: disable write to secure register 80 0 = OFF 1 = ON
15	0x0	READ79: disable read from secure register 79 0 = OFF 1 = ON

Bit	Reset	Description
14	0x0	WRITE79: disable write to secure register 79 0 = OFF 1 = ON
13	0x0	READ78: disable read from secure register 78 0 = OFF 1 = ON
12	0x0	WRITE78: disable write to secure register 78 0 = OFF 1 = ON
11	0x0	READ77: disable read from secure register 77 0 = OFF 1 = ON
10	0x0	WRITE77: disable write to secure register 77 0 = OFF 1 = ON
9	0x0	READ76: disable read from secure register 76 0 = OFF 1 = ON
8	0x0	WRITE76: disable write to secure register 76 0 = OFF 1 = ON
7	0x0	READ75: disable read from secure register 75 0 = OFF 1 = ON
6	0x0	WRITE75: disable write to secure register 75 0 = OFF 1 = ON
5	0x0	READ74: disable read from secure register 74 0 = OFF 1 = ON
4	0x0	WRITE74: disable write to secure register 74 0 = OFF 1 = ON
3	0x0	READ73: disable read from secure register 73 0 = OFF 1 = ON
2	0x0	WRITE73: disable write to secure register 73 0 = OFF 1 = ON
1	0x0	READ72: disable read from secure register 72 0 = OFF, 1 = ON 0 = OFF 1 = ON
0	0x0	WRITE72: disable write to secure register 72 0 = OFF, 1 = ON 0 = OFF 1 = ON

12.6.296 APBDEV_PMC_SEC_DISABLE7_0

Offset: 0x5bc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ103: disable read from secure register 103 0 = OFF 1 = ON
30	0x0	WRITE103: disable write to secure register 103 0 = OFF 1 = ON
29	0x0	READ102: disable read from secure register 102 0 = OFF 1 = ON

Bit	Reset	Description
28	0x0	WRITE102: disable write to secure register 102 0 = OFF 1 = ON
27	0x0	READ101: disable read from secure register 101 0 = OFF 1 = ON
26	0x0	WRITE101: disable write to secure register 101 0 = OFF 1 = ON
25	0x0	READ100: disable read from secure register 100 0 = OFF 1 = ON
24	0x0	WRITE100: disable write to secure register 100 0 = OFF 1 = ON
23	0x0	READ99: disable read from secure register 99 0 = OFF 1 = ON
22	0x0	WRITE99: disable write to secure register 99 0 = OFF 1 = ON
21	0x0	READ98: disable read from secure register 98 0 = OFF 1 = ON
20	0x0	WRITE98: disable write to secure register 98 0 = OFF 1 = ON
19	0x0	READ97: disable read from secure register 97 0 = OFF 1 = ON
18	0x0	WRITE97: disable write to secure register 97 0 = OFF 1 = ON
17	0x0	READ96: disable read from secure register 96 0 = OFF 1 = ON
16	0x0	WRITE96: disable write to secure register 96 0 = OFF 1 = ON
15	0x0	READ95: disable read from secure register 95 0 = OFF 1 = ON
14	0x0	WRITE95: disable write to secure register 95 0 = OFF 1 = ON
13	0x0	READ94: disable read from secure register 94 0 = OFF 1 = ON
12	0x0	WRITE94: disable write to secure register 94 0 = OFF 1 = ON
11	0x0	READ93: disable read from secure register 93 0 = OFF 1 = ON
10	0x0	WRITE93: disable write to secure register 93 0 = OFF 1 = ON
9	0x0	READ92: disable read from secure register 92 0 = OFF 1 = ON

Bit	Reset	Description
8	0x0	WRITE92: disable write to secure register 92 0 = OFF 1 = ON
7	0x0	READ91: disable read from secure register 91 0 = OFF 1 = ON
6	0x0	WRITE91: disable write to secure register 91 0 = OFF 1 = ON
5	0x0	READ90: disable read from secure register 90 0 = OFF 1 = ON
4	0x0	WRITE90: disable write to secure register 90 0 = OFF 1 = ON
3	0x0	READ89: disable read from secure register 89 0 = OFF 1 = ON
2	0x0	WRITE89: disable write to secure register 89 0 = OFF 1 = ON
1	0x0	READ88: disable read from secure register 88 0 = OFF, 1 = ON 0 = OFF 1 = ON
0	0x0	WRITE88: disable write to secure register 88 0 = OFF, 1 = ON 0 = OFF 1 = ON

12.6.297 APBDEV_PMC_SEC_DISABLE8_0

Offset: 0x5c0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ119: disable read from secure register 119 0 = OFF 1 = ON
30	0x0	WRITE119: disable write to secure register 119 0 = OFF 1 = ON
29	0x0	READ118: disable read from secure register 118 0 = OFF 1 = ON
28	0x0	WRITE118: disable write to secure register 118 0 = OFF 1 = ON
27	0x0	READ117: disable read from secure register 117 0 = OFF 1 = ON
26	0x0	WRITE117: disable write to secure register 117 0 = OFF 1 = ON
25	0x0	READ116: disable read from secure register 116 0 = OFF 1 = ON
24	0x0	WRITE116: disable write to secure register 116 0 = OFF 1 = ON
23	0x0	READ115: disable read from secure register115 0 = OFF 1 = ON

Bit	Reset	Description
22	0x0	WRITE115: disable write to secure register115 0 = OFF 1 = ON
21	0x0	READ114: disable read from secure register114 0 = OFF 1 = ON
20	0x0	WRITE114: disable write to secure register114 0 = OFF 1 = ON
19	0x0	READ113: disable read from secure register113 0 = OFF 1 = ON
18	0x0	WRITE113: disable write to secure register113 0 = OFF 1 = ON
17	0x0	READ112: disable read from secure register112 0 = OFF 1 = ON
16	0x0	WRITE112: disable write to secure register112 0 = OFF 1 = ON
15	0x0	READ111: disable read from secure register111 0 = OFF 1 = ON
14	0x0	WRITE111: disable write to secure register111 0 = OFF 1 = ON
13	0x0	READ110: disable read from secure register110 0 = OFF 1 = ON
12	0x0	WRITE110: disable write to secure register110 0 = OFF 1 = ON
11	0x0	READ109: disable read from secure register109 0 = OFF 1 = ON
10	0x0	WRITE109: disable write to secure register109 0 = OFF 1 = ON
9	0x0	READ108: disable read from secure register108 0 = OFF 1 = ON
8	0x0	WRITE108: disable write to secure register108 0 = OFF 1 = ON
7	0x0	READ107: disable read from secure register107 0 = OFF 1 = ON
6	0x0	WRITE107: disable write to secure register 107 0 = OFF 1 = ON
5	0x0	READ106: disable read from secure register 106 0 = OFF 1 = ON
4	0x0	WRITE106: disable write to secure register 106 0 = OFF 1 = ON
3	0x0	READ105: disable read from secure register 105 0 = OFF 1 = ON

Bit	Reset	Description
2	0x0	WRITE105: disable write to secure register 105 0 = OFF 1 = ON
1	0x0	READ104: disable read from secure register 104 0 = OFF, 1 = ON 0 = OFF 1 = ON
0	0x0	WRITE104: disable write to secure register 104 0 = OFF, 1 = ON 0 = OFF 1 = ON

12.6.298 APBDEV_PMC_SCRATCH56_0

Scratch register

Offset: 0x600 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH56: General purpose register storage

12.6.299 APBDEV_PMC_SCRATCH57_0

Scratch register

Offset: 0x604 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH57: General purpose register storage

12.6.300 APBDEV_PMC_SCRATCH58_0

Scratch register

Offset: 0x608 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH58: General purpose register storage

12.6.301 APBDEV_PMC_SCRATCH59_0

Scratch register

Offset: 0x60c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH59: General purpose register storage

12.6.302 APBDEV_PMC_SCRATCH60_0

Scratch register

Offset: 0x610 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH60: General purpose register storage

12.6.303 APBDEV_PMC_SCRATCH61_0

Scratch register

Offset: 0x614 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH61: General purpose register storage

12.6.304 APBDEV_PMC_SCRATCH62_0

Scratch register

Offset: 0x618 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH62: General purpose register storage

12.6.305 APBDEV_PMC_SCRATCH63_0

Scratch register

Offset: 0x61c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH63: General purpose register storage

12.6.306 APBDEV_PMC_SCRATCH64_0

Scratch register

Offset: 0x620 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH64: General purpose register storage

12.6.307 APBDEV_PMC_SCRATCH65_0

Scratch register

Offset: 0x624 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH65: General purpose register storage

12.6.308 APBDEV_PMC_SCRATCH66_0

Scratch register

Offset: 0x628 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH66: General purpose register storage

12.6.309 APBDEV_PMC_SCRATCH67_0

Scratch register

Offset: 0x62c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH67: General purpose register storage

12.6.310 APBDEV_PMC_SCRATCH68_0

Scratch Register

Offset: 0x630 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH68: General-purpose register storage

12.6.311 APBDEV_PMC_SCRATCH69_0

Scratch Register

Offset: 0x634 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH69: General-purpose register storage

12.6.312 APBDEV_PMC_SCRATCH70_0

Scratch Register

Offset: 0x638 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH70: General-purpose register storage

12.6.313 APBDEV_PMC_SCRATCH71_0

Scratch Register

Offset: 0x63c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH71: General-purpose register storage

12.6.314 APBDEV_PMC_SCRATCH72_0

Scratch Register

Offset: 0x640 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH72: General-purpose register storage

12.6.315 APBDEV_PMC_SCRATCH73_0

Scratch Register

Offset: 0x644 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH73: General-purpose register storage

12.6.316 APBDEV_PMC_SCRATCH74_0

Scratch Register

Offset: 0x648 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH74: General-purpose register storage

12.6.317 APBDEV_PMC_SCRATCH75_0

Scratch Register

Offset: 0x64c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH75: General-purpose register storage

12.6.318 APBDEV_PMC_SCRATCH76_0

Scratch Register

Offset: 0x650 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH76: General-purpose register storage

12.6.319 APBDEV_PMC_SCRATCH77_0

Scratch Register

Offset: 0x654 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH77: General-purpose register storage

12.6.320 APBDEV_PMC_SCRATCH78_0

Scratch Register

Offset: 0x658 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH78: General-purpose register storage

12.6.321 APBDEV_PMC_SCRATCH79_0

Scratch Register

Offset: 0x65c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH79: General-purpose register storage

12.6.322 APBDEV_PMC_SCRATCH80_0

Scratch Register

Offset: 0x660 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH80: General-purpose register storage

12.6.323 APBDEV_PMC_SCRATCH81_0

Scratch Register

Offset: 0x664 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH81: General-purpose register storage

12.6.324 APBDEV_PMC_SCRATCH82_0

Scratch Register

Offset: 0x668 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH82: General-purpose register storage

12.6.325 APBDEV_PMC_SCRATCH83_0

Scratch Register

Offset: 0x66c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH83: General-purpose register storage

12.6.326 APBDEV_PMC_SCRATCH84_0

Scratch Register

Offset: 0x670 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH84: General-purpose register storage

12.6.327 APBDEV_PMC_SCRATCH85_0

Scratch Register

Offset: 0x674 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH85: General-purpose register storage

12.6.328 APBDEV_PMC_SCRATCH86_0

Scratch Register

Offset: 0x678 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH86: General-purpose register storage

12.6.329 APBDEV_PMC_SCRATCH87_0

Scratch Register

Offset: 0x67c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH87: General-purpose register storage

12.6.330 APBDEV_PMC_SCRATCH88_0

Scratch Register

Offset: 0x680 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH88: General-purpose register storage

12.6.331 APBDEV_PMC_SCRATCH89_0

Scratch Register

Offset: 0x684 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH89: General-purpose register storage

12.6.332 APBDEV_PMC_SCRATCH90_0

Scratch Register

Offset: 0x688 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH90: General-purpose register storage

12.6.333 APBDEV_PMC_SCRATCH91_0

Scratch Register

Offset: 0x68c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH91: General-purpose register storage

12.6.334 APBDEV_PMC_SCRATCH92_0

Scratch Register

Offset: 0x690 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH92: General-purpose register storage

12.6.335 APBDEV_PMC_SCRATCH93_0

Scratch Register

Offset: 0x694 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH93: General-purpose register storage

12.6.336 APBDEV_PMC_SCRATCH94_0

Scratch Register

Offset: 0x698 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH94: General-purpose register storage

12.6.337 APBDEV_PMC_SCRATCH95_0

Scratch Register

Offset: 0x69c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH95: General-purpose register storage

12.6.338 APBDEV_PMC_SCRATCH96_0

Scratch Register

Offset: 0x6a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH96: General-purpose register storage

12.6.339 APBDEV_PMC_SCRATCH97_0

Scratch Register

Offset: 0x6a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH97: General-purpose register storage

12.6.340 APBDEV_PMC_SCRATCH98_0

Scratch Register

Offset: 0x6a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH98: General-purpose register storage

12.6.341 APBDEV_PMC_SCRATCH99_0

Scratch Register

Offset: 0x6ac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH99: General-purpose register storage

12.6.342 APBDEV_PMC_SCRATCH100_0

Scratch Register

Offset: 0x6b0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH100: General-purpose register storage

12.6.343 APBDEV_PMC_SCRATCH101_0

Scratch Register

Offset: 0x6b4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH101: General-purpose register storage

12.6.344 APBDEV_PMC_SCRATCH102_0

Scratch Register

Offset: 0x6b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH102: General-purpose register storage

12.6.345 APBDEV_PMC_SCRATCH103_0

Scratch Register

Offset: 0x6bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH103: General-purpose register storage

12.6.346 APBDEV_PMC_SCRATCH104_0

Scratch Register

Offset: 0x6c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH104: General-purpose register storage

12.6.347 APBDEV_PMC_SCRATCH105_0

Scratch Register

Offset: 0x6c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH105: General-purpose register storage

12.6.348 APBDEV_PMC_SCRATCH106_0

Scratch Register

Offset: 0x6c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH106: General-purpose register storage

12.6.349 APBDEV_PMC_SCRATCH107_0

Scratch Register

Offset: 0x6cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH107: General-purpose register storage

12.6.350 APBDEV_PMC_SCRATCH108_0

Scratch Register

Offset: 0x6d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH108: General-purpose register storage

12.6.351 APBDEV_PMC_SCRATCH109_0

Scratch Register

Offset: 0x6d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH109: General-purpose register storage

12.6.352 APBDEV_PMC_SCRATCH110_0

Scratch Register

Offset: 0x6d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH110: General-purpose register storage

12.6.353 APBDEV_PMC_SCRATCH111_0

Scratch Register

Offset: 0x6dc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH111: General-purpose register storage

12.6.354 APBDEV_PMC_SCRATCH112_0

Scratch Register

Offset: 0x6e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH112: General-purpose register storage

12.6.355 APBDEV_PMC_SCRATCH113_0

Scratch Register

Offset: 0x6e4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH113: General-purpose register storage

12.6.356 APBDEV_PMC_SCRATCH114_0

Scratch Register

Offset: 0x6e8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH114: General-purpose register storage

12.6.357 APBDEV_PMC_SCRATCH115_0

Scratch Register

Offset: 0x6ec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH115: General-purpose register storage

12.6.358 APBDEV_PMC_SCRATCH116_0

Scratch Register

Offset: 0x6f0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH116: General-purpose register storage

12.6.359 APBDEV_PMC_SCRATCH117_0

Scratch Register

Offset: 0x6f4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH117: General-purpose register storage

12.6.360 APBDEV_PMC_SCRATCH118_0

Scratch Register

Offset: 0x6f8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH118: General-purpose register storage

12.6.361 APBDEV_PMC_SCRATCH119_0

Scratch Register

Offset: 0x6fc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH119: General-purpose register storage

12.6.362 APBDEV_PMC_SCRATCH120 _0

Scratch register

Offset: 0x700 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH120: General purpose register storage

12.6.363 APBDEV_PMC_SCRATCH121_0

Scratch register

Offset: 0x704 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH121: General purpose register storage

12.6.364 APBDEV_PMC_SCRATCH122_0

Scratch register

Offset: 0x708 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH122: General purpose register storage

12.6.365 APBDEV_PMC_SCRATCH123_0

Scratch register

Offset: 0x70c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH123: General purpose register storage

12.6.366 APBDEV_PMC_SCRATCH124_0

Scratch register

Offset: 0x710 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH124: General purpose register storage

12.6.367 APBDEV_PMC_SCRATCH125_0

Scratch register

Offset: 0x714 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH125: General purpose register storage

12.6.368 APBDEV_PMC_SCRATCH126_0

Scratch register

Offset: 0x718 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH126: General purpose register storage

12.6.369 APBDEV_PMC_SCRATCH127_0

Scratch register

Offset: 0x71c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH127: General purpose register storage

12.6.370 APBDEV_PMC_SCRATCH128_0

Scratch register

Offset: 0x720 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH128: General purpose register storage

12.6.371 APBDEV_PMC_SCRATCH129_0

Scratch register

Offset: 0x724 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH129: General purpose register storage

12.6.372 APBDEV_PMC_SCRATCH130_0

Scratch register

Offset: 0x728 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH130: General purpose register storage

12.6.373 APBDEV_PMC_SCRATCH131_0

Scratch register

Offset: 0x72c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH131: General purpose register storage

12.6.374 APBDEV_PMC_SCRATCH132_0

Scratch register

Offset: 0x730 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH132: General purpose register storage

12.6.375 APBDEV_PMC_SCRATCH133_0

Scratch register

Offset: 0x734 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH133: General purpose register storage

12.6.376 APBDEV_PMC_SCRATCH134_0

Scratch register

Offset: 0x738 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH134: General purpose register storage

12.6.377 APBDEV_PMC_SCRATCH135_0

Scratch register

Offset: 0x73c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH135: General purpose register storage

12.6.378 APBDEV_PMC_SCRATCH136_0

Scratch register

Offset: 0x740 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH136: General purpose register storage

12.6.379 APBDEV_PMC_SCRATCH137_0

Scratch register

Offset: 0x744 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH137: General purpose register storage

12.6.380 APBDEV_PMC_SCRATCH138_0

Scratch register

Offset: 0x748 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH138: General purpose register storage

12.6.381 APBDEV_PMC_SCRATCH139_0

Scratch register

Offset: 0x74c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH139: General purpose register storage

12.6.382 APBDEV_PMC_SCRATCH140_0

Scratch register

Offset: 0x750 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH140: General purpose register storage

12.6.383 APBDEV_PMC_SCRATCH141_0

Scratch register

Offset: 0x754 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH141: General purpose register storage

12.6.384 APBDEV_PMC_SCRATCH142_0

Scratch register

Offset: 0x758 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH142: General purpose register storage

12.6.385 APBDEV_PMC_SCRATCH143_0

Scratch register

Offset: 0x75c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH143: General purpose register storage

12.6.386 APBDEV_PMC_SCRATCH144_0

Scratch register

Offset: 0x760 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH144: General purpose register storage

12.6.387 APBDEV_PMC_SCRATCH145_0

Scratch register

Offset: 0x764 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH145: General purpose register storage

12.6.388 APBDEV_PMC_SCRATCH146_0

Scratch register

Offset: 0x768 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH146: General purpose register storage

12.6.389 APBDEV_PMC_SCRATCH147_0

Scratch register

Offset: 0x76c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH147: General purpose register storage

12.6.390 APBDEV_PMC_SCRATCH148_0

Scratch register

Offset: 0x770 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH148: General purpose register storage

12.6.391 APBDEV_PMC_SCRATCH149_0

Scratch register

Offset: 0x774 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH149: General purpose register storage

12.6.392 APBDEV_PMC_SCRATCH150_0

Scratch register

Offset: 0x778 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH150: General purpose register storage

12.6.393 APBDEV_PMC_SCRATCH151_0

Scratch register

Offset: 0x77c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH151: General purpose register storage

12.6.394 APBDEV_PMC_SCRATCH152_0

Scratch register

Offset: 0x780 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH152: General purpose register storage

12.6.395 APBDEV_PMC_SCRATCH153_0

Scratch register

Offset: 0x784 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH153: General purpose register storage

12.6.396 APBDEV_PMC_SCRATCH154_0

Scratch register

Offset: 0x788 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH154: General purpose register storage

12.6.397 APBDEV_PMC_SCRATCH155_0

Scratch register

Offset: 0x78c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH155: General purpose register storage

12.6.398 APBDEV_PMC_SCRATCH156_0

Scratch register

Offset: 0x790 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH156: General purpose register storage

12.6.399 APBDEV_PMC_SCRATCH157_0

Scratch register

Offset: 0x794 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH157: General purpose register storage

12.6.400 APBDEV_PMC_SCRATCH158_0

Scratch register

Offset: 0x798 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH158: General purpose register storage

12.6.401 APBDEV_PMC_SCRATCH159_0

Scratch register

Offset: 0x79c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH159: General purpose register storage

12.6.402 APBDEV_PMC_SCRATCH160_0

Scratch register

Offset: 0x7a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH160: General purpose register storage

12.6.403 APBDEV_PMC_SCRATCH161_0

Scratch register

Offset: 0x7a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH161: General purpose register storage

12.6.404 APBDEV_PMC_SCRATCH162_0

Scratch register

Offset: 0x7a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH162: General purpose register storage

12.6.405 APBDEV_PMC_SCRATCH163_0

Scratch register

Offset: 0x7ac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH163: General purpose register storage

12.6.406 APBDEV_PMC_SCRATCH164_0

Scratch register

Offset: 0x7b0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH164: General purpose register storage

12.6.407 APBDEV_PMC_SCRATCH165_0

Scratch register

Offset: 0x7b4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH165: General purpose register storage

12.6.408 APBDEV_PMC_SCRATCH166_0

Scratch register

Offset: 0x7b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH166: General purpose register storage

12.6.409 APBDEV_PMC_SCRATCH167_0

Scratch register

Offset: 0x7bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH167: General purpose register storage

12.6.410 APBDEV_PMC_SCRATCH168_0

Scratch register

Offset: 0x7c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH168: General purpose register storage

12.6.411 APBDEV_PMC_SCRATCH169_0

Scratch register

Offset: 0x7c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH169: General purpose register storage

12.6.412 APBDEV_PMC_SCRATCH170_0

Scratch register

Offset: 0x7c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH170: General purpose register storage

12.6.413 APBDEV_PMC_SCRATCH171_0

Scratch register

Offset: 0x7cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH171: General purpose register storage

12.6.414 APBDEV_PMC_SCRATCH172_0

Scratch register

Offset: 0x7d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH172: General purpose register storage

12.6.415 APBDEV_PMC_SCRATCH173_0

Scratch register

Offset: 0x7d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH173: General purpose register storage

12.6.416 APBDEV_PMC_SCRATCH174_0

Scratch register

Offset: 0x7d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH174: General purpose register storage

12.6.417 APBDEV_PMC_SCRATCH175_0

Scratch register

Offset: 0x7dc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH175: General purpose register storage

12.6.418 APBDEV_PMC_SCRATCH176_0

Scratch register

Offset: 0x7e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH176: General purpose register storage

12.6.419 APBDEV_PMC_SCRATCH177_0

Scratch register

Offset: 0x7e4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH177: General purpose register storage

12.6.420 APBDEV_PMC_SCRATCH178_0

Scratch register

Offset: 0x7e8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH178: General purpose register storage

12.6.421 APBDEV_PMC_SCRATCH179_0

Scratch register

Offset: 0x7ec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH179: General purpose register storage

12.6.422 APBDEV_PMC_SCRATCH180_0

Scratch register

Offset: 0x7f0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH180: General purpose register storage

12.6.423 APBDEV_PMC_SCRATCH181_0

Scratch register

Offset: 0x7f4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH181: General purpose register storage

12.6.424 APBDEV_PMC_SCRATCH182_0

Scratch register

Offset: 0x7f8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH182: General purpose register storage

12.6.425 APBDEV_PMC_SCRATCH183_0

Scratch register

Offset: 0x7fc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH183: General purpose register storage

12.6.426 APBDEV_PMC_SCRATCH184_0

Scratch register

Offset: 0x800 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH184: General purpose register storage

12.6.427 APBDEV_PMC_SCRATCH185_0

Scratch register

Offset: 0x804 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH185: General purpose register storage

12.6.428 APBDEV_PMC_SCRATCH186_0

Scratch register

Offset: 0x808 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH186: General purpose register storage

12.6.429 APBDEV_PMC_SCRATCH187_0

Scratch register

Offset: 0x80c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH187: General purpose register storage

12.6.430 APBDEV_PMC_SCRATCH188_0

Scratch register

Offset: 0x810 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH188: General purpose register storage

12.6.431 APBDEV_PMC_SCRATCH189_0

Scratch register

Offset: 0x814 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH189: General purpose register storage

12.6.432 APBDEV_PMC_SCRATCH190_0

Scratch register

Offset: 0x818 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH190: General purpose register storage

12.6.433 APBDEV_PMC_SCRATCH191_0

Scratch register

Offset: 0x81c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH191: General purpose register storage

12.6.434 APBDEV_PMC_SCRATCH192_0

Scratch register

Offset: 0x820 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH192: General purpose register storage

12.6.435 APBDEV_PMC_SCRATCH193_0

Scratch register

Offset: 0x824 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH193: General purpose register storage

12.6.436 APBDEV_PMC_SCRATCH194_0

Scratch register

Offset: 0x828 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH194: General purpose register storage

12.6.437 APBDEV_PMC_SCRATCH195_0

Scratch register

Offset: 0x82c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH195: General purpose register storage

12.6.438 APBDEV_PMC_SCRATCH196_0

Scratch register

Offset: 0x830 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH196: General purpose register storage

12.6.439 APBDEV_PMC_SCRATCH197_0

Scratch register

Offset: 0x834 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH197: General purpose register storage

12.6.440 APBDEV_PMC_SCRATCH198_0

Scratch register

Offset: 0x838 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH198: General purpose register storage

12.6.441 APBDEV_PMC_SCRATCH199_0

Scratch register

Offset: 0x83c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH199: General purpose register storage

12.6.442 APBDEV_PMC_SCRATCH200_0

Scratch register

Offset: 0x840 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH200: General purpose register storage

12.6.443 APBDEV_PMC_SCRATCH201_0

Scratch register

Offset: 0x844 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH201: General purpose register storage

12.6.444 APBDEV_PMC_SCRATCH202_0

Scratch register

Offset: 0x848 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH202: General purpose register storage

12.6.445 APBDEV_PMC_SCRATCH203_0

Scratch register

Offset: 0x84c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH203: General purpose register storage

12.6.446 APBDEV_PMC_SCRATCH204_0

Scratch register

Offset: 0x850 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH204: General purpose register storage

12.6.447 APBDEV_PMC_SCRATCH205_0

Scratch register

Offset: 0x854 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH205: General purpose register storage

12.6.448 APBDEV_PMC_SCRATCH206_0

Scratch register

Offset: 0x858 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH206: General purpose register storage

12.6.449 APBDEV_PMC_SCRATCH207_0

Scratch register

Offset: 0x85c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH207: General purpose register storage

12.6.450 APBDEV_PMC_SCRATCH208_0

Scratch register

Offset: 0x860 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH208: General purpose register storage

12.6.451 APBDEV_PMC_SCRATCH209_0

Scratch register

Offset: 0x864 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH209: General purpose register storage

12.6.452 APBDEV_PMC_SCRATCH210_0

Scratch register

Offset: 0x868 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH210: General purpose register storage

12.6.453 APBDEV_PMC_SCRATCH211_0

Scratch register

Offset: 0x86c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH211: General purpose register storage

12.6.454 APBDEV_PMC_SCRATCH212_0

Scratch register

Offset: 0x870 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH212: General purpose register storage

12.6.455 APBDEV_PMC_SCRATCH213_0

Scratch register

Offset: 0x874 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH213: General purpose register storage

12.6.456 APBDEV_PMC_SCRATCH214_0

Scratch register

Offset: 0x878 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH214: General purpose register storage

12.6.457 APBDEV_PMC_SCRATCH215_0

Scratch register

Offset: 0x87c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH215: General purpose register storage

12.6.458 APBDEV_PMC_SCRATCH216_0

Scratch register

Offset: 0x880 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH216: General purpose register storage

12.6.459 APBDEV_PMC_SCRATCH217_0

Scratch register

Offset: 0x884 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH217: General purpose register storage

12.6.460 APBDEV_PMC_SCRATCH218_0

Scratch register

Offset: 0x888 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH218: General purpose register storage

12.6.461 APBDEV_PMC_SCRATCH219_0

Scratch register

Offset: 0x88c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH219: General purpose register storage

12.6.462 APBDEV_PMC_SCRATCH220_0

Scratch register

Offset: 0x890 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH220: General purpose register storage

12.6.463 APBDEV_PMC_SCRATCH221_0

Scratch register

Offset: 0x894 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH221: General purpose register storage

12.6.464 APBDEV_PMC_SCRATCH222_0

Scratch register

Offset: 0x898 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH222: General purpose register storage

12.6.465 APBDEV_PMC_SCRATCH223_0

Scratch register

Offset: 0x89c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH223: General purpose register storage

12.6.466 APBDEV_PMC_SCRATCH224_0

Scratch register

Offset: 0x8a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH224: General purpose register storage

12.6.467 APBDEV_PMC_SCRATCH225_0

Scratch register

Offset: 0x8a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH225: General purpose register storage

12.6.468 APBDEV_PMC_SCRATCH226_0

Scratch register

Offset: 0x8a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH226: General purpose register storage

12.6.469 APBDEV_PMC_SCRATCH227_0

Scratch register

Offset: 0x8ac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH227: General purpose register storage

12.6.470 APBDEV_PMC_SCRATCH228_0

Scratch register

Offset: 0x8b0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH228: General purpose register storage

12.6.471 APBDEV_PMC_SCRATCH229_0

Scratch register

Offset: 0x8b4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH229: General purpose register storage

12.6.472 APBDEV_PMC_SCRATCH230_0

Scratch register

Offset: 0x8b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH230: General purpose register storage

12.6.473 APBDEV_PMC_SCRATCH231_0

Scratch register

Offset: 0x8bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH231: General purpose register storage

12.6.474 APBDEV_PMC_SCRATCH232_0

Scratch register

Offset: 0x8c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH232: General purpose register storage

12.6.475 APBDEV_PMC_SCRATCH233_0

Scratch register

Offset: 0x8c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH233: General purpose register storage

12.6.476 APBDEV_PMC_SCRATCH234_0

Scratch register

Offset: 0x8c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH234: General purpose register storage

12.6.477 APBDEV_PMC_SCRATCH235_0

Scratch register

Offset: 0x8cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH235: General purpose register storage

12.6.478 APBDEV_PMC_SCRATCH236_0

Scratch register

Offset: 0x8d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH236: General purpose register storage

12.6.479 APBDEV_PMC_SCRATCH237_0

Scratch register

Offset: 0x8d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH237: General purpose register storage

12.6.480 APBDEV_PMC_SCRATCH238_0

Scratch register

Offset: 0x8d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH238: General purpose register storage

12.6.481 APBDEV_PMC_SCRATCH239_0

Scratch register

Offset: 0x8dc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH239: General purpose register storage

12.6.482 APBDEV_PMC_SCRATCH240_0

Scratch register

Offset: 0x8e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH240: General purpose register storage

12.6.483 APBDEV_PMC_SCRATCH241_0

Scratch register

Offset: 0x8e4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH241: General purpose register storage

12.6.484 APBDEV_PMC_SCRATCH242_0

Scratch register

Offset: 0x8e8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH242: General purpose register storage

12.6.485 APBDEV_PMC_SCRATCH243_0

Scratch register

Offset: 0x8ec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH243: General purpose register storage

12.6.486 APBDEV_PMC_SCRATCH244_0

Scratch register

Offset: 0x8f0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

12.6.487 APBDEV_PMC_SCRATCH245_0

Bit	Reset	Description
31:0	X	SCRATCH244: General purpose register storage

Scratch register

Offset: 0x8f4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH245: General purpose register storage

12.6.488 APBDEV_PMC_SCRATCH246_0

Scratch register

Offset: 0x8f8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH246: General purpose register storage

12.6.489 APBDEV_PMC_SCRATCH247_0

Scratch register

Offset: 0x8fc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH247: General purpose register storage

12.6.490 APBDEV_PMC_SCRATCH248_0

Scratch register

Offset: 0x900 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH248: General purpose register storage

12.6.491 APBDEV_PMC_SCRATCH249_0

Scratch register

Offset: 0x904 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH249: General purpose register storage

12.6.492 APBDEV_PMC_SCRATCH250_0

Scratch register

Offset: 0x908 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH250: General purpose register storage

12.6.493 APBDEV_PMC_SCRATCH251_0

Scratch register

Offset: 0x90c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH251: General purpose register storage

12.6.494 APBDEV_PMC_SCRATCH252_0

Scratch register

Offset: 0x910 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH252: General purpose register storage

12.6.495 APBDEV_PMC_SCRATCH253_0

Scratch register

Offset: 0x914 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH253: General purpose register storage

12.6.496 APBDEV_PMC_SCRATCH254_0

Scratch register

Offset: 0x918 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH254: General purpose register storage

12.6.497 APBDEV_PMC_SCRATCH255_0

Scratch register

Offset: 0x91c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH255: General purpose register storage

12.6.498 APBDEV_PMC_SCRATCH256_0

Scratch register

Offset: 0x920 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH256: General purpose register storage

12.6.499 APBDEV_PMC_SCRATCH257_0

Scratch register

Offset: 0x924 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH257: General purpose register storage

12.6.500 APBDEV_PMC_SCRATCH258_0

Scratch register

Offset: 0x928 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH258: General purpose register storage

12.6.501 APBDEV_PMC_SCRATCH259_0

Scratch register

Offset: 0x92c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH259: General purpose register storage

12.6.502 APBDEV_PMC_SCRATCH260_0

Scratch register

Offset: 0x930 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH260: General purpose register storage

12.6.503 APBDEV_PMC_SCRATCH261_0

Scratch register

Offset: 0x934 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH261: General purpose register storage

12.6.504 APBDEV_PMC_SCRATCH262_0

Scratch register

Offset: 0x938 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH262: General purpose register storage

12.6.505 APBDEV_PMC_SCRATCH263_0

Scratch register

Offset: 0x93c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH263: General purpose register storage

12.6.506 APBDEV_PMC_SCRATCH264_0

Scratch register

Offset: 0x940 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH264: General purpose register storage

12.6.507 APBDEV_PMC_SCRATCH265_0

Scratch register

Offset: 0x944 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH265: General purpose register storage

12.6.508 APBDEV_PMC_SCRATCH266_0

Scratch register

Offset: 0x948 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH266: General purpose register storage

12.6.509 APBDEV_PMC_SCRATCH267_0

Scratch register

Offset: 0x94c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH267: General purpose register storage

12.6.510 APBDEV_PMC_SCRATCH268_0

Scratch register

Offset: 0x950 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH268: General purpose register storage

12.6.511 APBDEV_PMC_SCRATCH269_0

Scratch register

Offset: 0x954 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH269: General purpose register storage

12.6.512 APBDEV_PMC_SCRATCH270_0

Scratch register

Offset: 0x958 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH270: General purpose register storage

12.6.513 APBDEV_PMC_SCRATCH271_0

Scratch register

Offset: 0x95c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH271: General purpose register storage

12.6.514 APBDEV_PMC_SCRATCH272_0

Scratch register

Offset: 0x960 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH272: General purpose register storage

12.6.515 APBDEV_PMC_SCRATCH273_0

Scratch register

Offset: 0x964 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH273: General purpose register storage

12.6.516 APBDEV_PMC_SCRATCH274_0

Scratch register

Offset: 0x968 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH274: General purpose register storage

12.6.517 APBDEV_PMC_SCRATCH275_0

Scratch register

Offset: 0x96c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH275: General purpose register storage

12.6.518 APBDEV_PMC_SCRATCH276_0

Scratch register

Offset: 0x970 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH276: General purpose register storage

12.6.519 APBDEV_PMC_SCRATCH277_0

Scratch register

Offset: 0x974 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH277: General purpose register storage

12.6.520 APBDEV_PMC_SCRATCH278_0

Scratch register

Offset: 0x978 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH278: General purpose register storage

12.6.521 APBDEV_PMC_SCRATCH279_0

Scratch register

Offset: 0x97c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH279: General purpose register storage

12.6.522 APBDEV_PMC_SCRATCH280_0

Scratch register

Offset: 0x980 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH280: General purpose register storage

12.6.523 APBDEV_PMC_SCRATCH281_0

Scratch register

Offset: 0x984 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH281: General purpose register storage

12.6.524 APBDEV_PMC_SCRATCH282_0

Scratch register

Offset: 0x988 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH282: General purpose register storage

12.6.525 APBDEV_PMC_SCRATCH283_0

Scratch register

Offset: 0x98c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH283: General purpose register storage

12.6.526 APBDEV_PMC_SCRATCH284_0

Scratch register

Offset: 0x990 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH284: General purpose register storage

12.6.527 APBDEV_PMC_SCRATCH285_0

Scratch register

Offset: 0x994 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH285: General purpose register storage

12.6.528 APBDEV_PMC_SCRATCH286_0

Scratch register

Offset: 0x998 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH286: General purpose register storage

12.6.529 APBDEV_PMC_SCRATCH287_0

Scratch register

Offset: 0x99c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH287: General purpose register storage

12.6.530 APBDEV_PMC_SCRATCH288_0

Scratch register

Offset: 0x9a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH288: General purpose register storage

12.6.531 APBDEV_PMC_SCRATCH289_0

Scratch register

Offset: 0x9a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH289: General purpose register storage

12.6.532 APBDEV_PMC_SCRATCH290_0

Scratch register

Offset: 0x9a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH290: General purpose register storage

12.6.533 APBDEV_PMC_SCRATCH291_0

Scratch register

Offset: 0x9ac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH291: General purpose register storage

12.6.534 APBDEV_PMC_SCRATCH292_0

Scratch register

Offset: 0x9b0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH292: General purpose register storage

12.6.535 APBDEV_PMC_SCRATCH293_0

Scratch register

Offset: 0x9b4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH293: General purpose register storage

12.6.536 APBDEV_PMC_SCRATCH294_0

Scratch register

Offset: 0x9b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH294: General purpose register storage

12.6.537 APBDEV_PMC_SCRATCH295_0

Scratch register

Offset: 0x9bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH295: General purpose register storage

12.6.538 APBDEV_PMC_SCRATCH296_0

Scratch register

Offset: 0x9c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH296: General purpose register storage

12.6.539 APBDEV_PMC_SCRATCH297_0

Scratch register

Offset: 0x9c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH297: General purpose register storage

12.6.540 APBDEV_PMC_SCRATCH298_0

Scratch register

Offset: 0x9c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH298: General purpose register storage

12.6.541 APBDEV_PMC_SCRATCH299_0

Scratch register

Offset: 0x9cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH299: General purpose register storage

12.6.542 APBDEV_PMC_SECURE_SCRATCH80_0

Secure scratch register

Reserved

Offset: 0xa98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH80

12.6.543 APBDEV_PMC_SECURE_SCRATCH81_0

Secure scratch register

Offset: 0xa9c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH81

12.6.544 APBDEV_PMC_SECURE_SCRATCH82_0

Secure scratch register

Offset: 0xaa0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH82

12.6.545 APBDEV_PMC_SECURE_SCRATCH83_0

Secure scratch register

Offset: 0xaa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH83

12.6.546 APBDEV_PMC_SECURE_SCRATCH84_0

Secure scratch register

Offset: 0xaa8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH84

12.6.547 APBDEV_PMC_SECURE_SCRATCH85_0

Secure scratch register

Offset: 0xaac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH85

12.6.548 APBDEV_PMC_SECURE_SCRATCH86_0

Secure scratch register

Offset: 0xab0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH86

12.6.549 APBDEV_PMC_SECURE_SCRATCH87_0

Secure scratch register

Offset: 0xab4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH87

12.6.550 APBDEV_PMC_SECURE_SCRATCH88_0

Secure scratch register

Offset: 0xab8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH88

12.6.551 APBDEV_PMC_SECURE_SCRATCH89_0

Secure scratch register

Offset: 0xabc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH89

12.6.552 APBDEV_PMC_SECURE_SCRATCH90_0

Secure scratch register

Offset: 0xac0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH90

12.6.553 APBDEV_PMC_SECURE_SCRATCH91_0

Secure scratch register

Offset: 0xac4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH91

12.6.554 APBDEV_PMC_SECURE_SCRATCH92_0

Secure scratch register

Offset: 0xac8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH92

12.6.555 APBDEV_PMC_SECURE_SCRATCH93_0

Secure scratch register

Offset: 0xacc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH93

12.6.556 APBDEV_PMC_SECURE_SCRATCH94_0

Secure scratch register

Offset: 0xad0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH94

12.6.557 APBDEV_PMC_SECURE_SCRATCH95_0

Secure scratch register

Offset: 0xad4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH95

12.6.558 APBDEV_PMC_SECURE_SCRATCH96_0

Secure scratch register

Offset: 0xad8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH96

12.6.559 APBDEV_PMC_SECURE_SCRATCH97_0

Secure scratch register

Offset: 0xadc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH97

12.6.560 APBDEV_PMC_SECURE_SCRATCH98_0

Secure scratch register

Offset: 0xae0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH98

12.6.561 APBDEV_PMC_SECURE_SCRATCH99_0

Secure scratch register

Offset: 0xae4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH98

12.6.562 APBDEV_PMC_SECURE_SCRATCH100_0

Secure scratch register

Offset: 0xae8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH100

12.6.563 APBDEV_PMC_SECURE_SCRATCH101_0

Secure scratch register

Offset: 0xaec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH101

12.6.564 APBDEV_PMC_SECURE_SCRATCH102_0

Secure scratch register

Offset: 0xaf0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH102

12.6.565 APBDEV_PMC_SECURE_SCRATCH103_0

Secure scratch register

Offset: 0xaf4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH103

12.6.566 APBDEV_PMC_SECURE_SCRATCH104_0

Secure scratch register

Offset: 0xaf8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH104

12.6.567 APBDEV_PMC_SECURE_SCRATCH105_0

Secure scratch register

Offset: 0xafc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH105

12.6.568 APBDEV_PMC_SECURE_SCRATCH106_0

Secure scratch register

Offset: 0xb00 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH106

12.6.569 APBDEV_PMC_SECURE_SCRATCH107_0

Secure scratch register

Offset: 0xb04 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH107

12.6.570 APBDEV_PMC_SECURE_SCRATCH108_0

Secure scratch register

Offset: 0xb08 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH108

12.6.571 APBDEV_PMC_SECURE_SCRATCH109_0

Secure scratch register

Offset: 0xb0c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH109

12.6.572 APBDEV_PMC_SECURE_SCRATCH110_0

Secure scratch register

Offset: 0xb10 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH110

12.6.573 APBDEV_PMC_SECURE_SCRATCH111_0

Secure scratch register

Offset: 0xb14 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH111

12.6.574 APBDEV_PMC_SECURE_SCRATCH112_0

Secure scratch register

Offset: 0xb18 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH112

12.6.575 APBDEV_PMC_SECURE_SCRATCH113_0

Secure scratch register

Offset: 0xb1c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH113

12.6.576 APBDEV_PMC_SECURE_SCRATCH114_0

Secure scratch register

Offset: 0xb20 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH114

12.6.577 APBDEV_PMC_SECURE_SCRATCH115_0

Secure scratch register

Offset: 0xb24 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH115

12.6.578 APBDEV_PMC_SECURE_SCRATCH116_0

Secure scratch register

Offset: 0xb28 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH116

12.6.579 APBDEV_PMC_SECURE_SCRATCH117_0

Secure scratch register

Offset: 0xb2c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH117

12.6.580 APBDEV_PMC_SECURE_SCRATCH118_0

Secure scratch register

Offset: 0xb30 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH118

12.6.581 APBDEV_PMC_SECURE_SCRATCH119_0

Secure scratch register

Offset: 0xb34 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH119

12.7 PMC Counter 0 Registers

Refer to Section 1.4: Reading Register Tables in the Introduction section for the register table protocol as well as recommendations for accessing registers.

12.7.1 SYSCTR0_CNTCR_0

The control registers are read/written by secure accesses only except the CNTFID0 and COUNTERID11-0 registers, which can be written only once by secure or non-secure access.

The CNTFID0 and COUNTERID11-0 registers are read-only; however, to initialize them at boot (by the ARM7 or main CPU), they are defined as write once.

A non-secure write to the control registers is ignored. A non-secure read to the control registers returns all 1s.

Counter Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx00)

Bit	Reset	Description
8	0x0	FCREQ: Requested frequency modes table entry. This bit is not used.
1	0x0	HDBG: Halt-on-debug. Controls whether a Halt-on-debug signal halts the system counter or not. 0: System counter ignores Halt-on-debug.1: Asserted Halt-on-debug signal halts the system counter update. 0 = DISABLE 1 = ENABLE
0	0x0	EN: Enables the counter. When this bit is written to '1', then the TSC is loaded with the CNTCVx register's value, and starts incrementing from that value. When this bit is written to '0', the TSC halts counting and stays at that value. 0: System counter disabled 1: System counter enabled. 0 = DISABLE 1 = ENABLE

12.7.2 SYSCTR0_CNTSR_0

Counter Status Register

Offset: 0x4 | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	FCREQ: Frequency change acknowledge. This bit is a copy of the value of the CNTCR[FCREQ] bit.
1	X	HDBG: Indicates whether or not the counter is halted because the Halt-on-Debug signal is asserted: 0: Counter is not halted. 1: Counter is halted. 0 = DISABLE 1 = ENABLE
0	X	RESERVED: Reserved, no impact.

12.7.3 SYSCTR0_CNTCV0_0

Counter Count Value [31:0] Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CV: Counter value [31:0]. A read of this register provides the TSC[31:0] value at the time of the read. When CNTCR[EN]=0, a write to this register is used to initialize the TSC[31:0] value. When CNTCR[EN]=1, a write to this register has an unpredictable behavior.

12.7.4 SYSCTR0_CNTCV1_0

Counter Count Value[63:32] Register

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CV: Counter value [63:32]. A read of this register provides the TSC[63:32] value at the time of the read. When CNTCR[EN]=0, a write to this register is used to initialize the TSC[63:32] value. When CNTCR[EN]=1, a write to this register has an unpredictable behavior.

12.7.5 SYSCTR0_CNTFID0_0

Frequency Table Entry Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00b71b00 (0b00000000101101110001101100000000)

Bit	Reset	Description
31:0	0xb71b00	FV: Counter frequency value in Hz. The default is set to 12 MHz. This register can be written only once.

12.7.6 SYSCTR0_CNTFID1_0

Frequency Table End Marker

Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	FV: Counter frequency value end marker (all 0s, read-only).

12.7.7 SYSCTR0_COUNTERID4_0

The COUNTERID11-0 registers are read-only; however, to initialize them at boot (by the ARM7™ or main CPU), they are defined as write once.

Offset: 0xfd0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

12.7.8 SYSCTR0_COUNTERID5_0

Offset: 0xfd4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

12.7.9 SYSCTR0_COUNTERID6_0

Offset: 0xfd8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

12.7.10 SYSCTR0_COUNTERID7_0

Offset: 0xfdc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

12.7.11 SYSCTR0_COUNTERID0_0

Offset: 0xfe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

12.7.12 SYSCTR0_COUNTERID1_0

Offset: 0xfe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

12.7.13 SYSCTR0_COUNTERID2_0

Offset: 0xfe8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

12.7.14 SYSCTR0_COUNTERID3_0

Offset: 0xfec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Peripheral ID value

12.7.15 SYSCTR0_COUNTERID8_0

Offset: 0xff0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

12.7.16 SYSCTR0_COUNTERID9_0

Offset: 0xff4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

12.7.17 SYSCTR0_COUNTERID10_0

Offset: 0xff8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

12.7.18 SYSCTR0_COUNTERID11_0

Offset: 0xffc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

12.8 PMC Counter 1 Registers

Refer to Section 1.4: Reading Register Tables in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The status registers are some control registers readable via the CNTReadBase base address by secure or non-secure accesses.

These are the same physical registers that are described in the Control Register section.

12.8.1 SYSCTR1_CNTCV0_0

Counter Count Value [31:0] Register

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CV: Counter value [31:0]

12.8.2 SYSCTR1_CNTCV1_0

Counter Count Value [63:32] Register

Offset: 0xc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CV: Counter value [63:32]

12.8.3 SYSCTR1_COUNTERID4_0

Offset: 0xfd0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value.

12.8.4 SYSCTR1_COUNTERID5_0

Offset: 0xfd4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

12.8.5 SYSCTR1_COUNTERID6_0

Offset: 0xfd8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

12.8.6 SYSCTR1_COUNTERID7_0

Offset: 0xfdc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED: Reserved. A read will return all 0s.

12.8.7 SYSCTR1_COUNTERID0_0

Offset: 0xfe0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

12.8.8 SYSCTR1_COUNTERID1_0

Offset: 0xfe4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

12.8.9 SYSCTR1_COUNTERID2_0

Offset: 0xfe8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

12.8.10 SYSCTR1_COUNTERID3_0

Offset: 0xfec | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Peripheral ID value

12.8.11 SYSCTR1_COUNTERID8_0

Offset: 0xff0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Component ID value

12.8.12 SYSCTR1_COUNTERID9_0

Offset: 0xff4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Component ID value

12.8.13 SYSCTR1_COUNTERID10_0

Offset: 0xff8 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: Component ID value

12.8.14 SYSCTR1_COUNTERID11_0

Offset: 0xffc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: Component ID value

12.9 Secure Boot Registers

12.9.1 SB_CSR_0

Secure Boot Control Status Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxx000000000000xx01)

Bit	R/W	Reset	Description
15:12	RW	0x0	COT_FAIL_COUNT: Chain-of-Trust Fail Count
11:8	RW	0x0	SWDM_FAIL_COUNT: Secure Watchdog Monitor fail Count
7	RW	0x0	SWDM_ENABLE: Secure Watchdog Monitor Enable 1 = ENABLE 0 = DISABLE
6	RW	0x0	HANG: Secure Boot Hang 1 = ENABLE 0 = DISABLE
5	RW	0x0	RESERVED_1: Reserved
4	RW	0x0	PIROM_DISABLE: Protected iROM Disable 0 = ENABLE 1 = DISABLE
3:2	RO	X	Rsrvd_31: Reserved
1	RW	0x0	NS_RST_VEC_WR_DIS: Non-secure reset vector write disable 0 = ENABLE 1 = DISABLE
0	RW	0x1	SECURE_BOOT_FLAG: 1 = Booted into Secure Mode 1 = ENABLE 0 = DISABLE

12.9.2 SB_PIROM_START_0

This specifies the offset from the start of the Boot ROM to the protected Region. This register is only programmable while in Secure_Mode (SECURE_BOOT_FLAG above == 1)

The lower 7 bits (6:0) are not significant and are assumed to be zero.

Secure Boot Protected ROM Start

Offset: 0x4 | Read/Write: R/W | Reset: 0x00001000 (0b00000000000000000001000000000000)

Bit	Reset	Description
31:0	0x1000	PROTECTED_ROM_START: PROTECTED_ROM_START

12.9.3 SB_PFCFG_0

Secure Boot Processor Feature Configuration Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx000000000xxx00000xx00xxxx)

Bit	Reset	Description
24:16	0x0	RESERVED_2: Reserved
12:8	0x0	RESERVED_3: Reserved
5	0x0	CP15SDISABLE: Disable access to system control registers 1 = DISABLE 0 = ENABLE
4	0x0	CFGSDISABLE: Disable access to secure control registers 1 = DISABLE 0 = ENABLE

12.9.4 SB_SECURE_SPAREREG_0_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_0

12.9.5 SB_SECURE_SPAREREG_1_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_1

12.9.6 SB_SECURE_SPAREREG_2_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_2

12.9.7 SB_SECURE_SPAREREG_3_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_3

12.9.8 SB_SECURE_SPAREREG_4_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_4

12.9.9 SB_SECURE_SPAREREG_5_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_5

12.9.10 SB_SECURE_SPAREREG_6_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_6

12.9.11 SB_SECURE_SPAREREG_7_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECURE_SPAREREG_7

12.9.12 SB_AA64_RESET_LOW_0

Two secure boot registers control the AA64 reset behavior of the ARM Cortex-A57, called AA64_RESET_.

For security reasons, these two registers have a protection similar to the AA32 CPU reset vector, including sharing the same sticky bit as AA32 CPU reset. When the sticky bit is set, all these registers become TrustZone secure for write accesses; i.e., they can only be modified by secure writes.

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	AA64_RESET_LOW

12.9.13 SB_AA64_RESET_HIGH_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	AA64_RESET_HIGH

12.10 EVP Register

12.10.1 EVP_RESET_VECTOR_0

The disable for this pointer is bit 1, NS_RST_VEC_WR_DIS, in the SB_CSR_0 register.

Offset: 0x0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RESET_VECTOR: RESET Exception Vector Pointer

12.10.2 EVP_CPU_RESET_VECTOR_0

Offset: 0x100 I Read/Write: R/W I Reset: 0x00100000 (0b00000000000100000000000000000000)

Bit	Reset	Description
31:0	0x100000	CPU_RESET_VECTOR: RESET Exception Vector Pointer

CHAPTER 13: BOOT PROCESS

13.1 TSense Reset Handler

The TSense reset is handled at the beginning of the Boot ROM secure ROM entry. The PLLP and PMC clocks are available for this reset.

The SCRATCH54 and SCRATCH55 registers deal with PMIC board-specific support (these are defined in Chapter 12: Power Management Controller). The PMIC driver configures them. The I^2C controller is used to communicate to the PMIC.

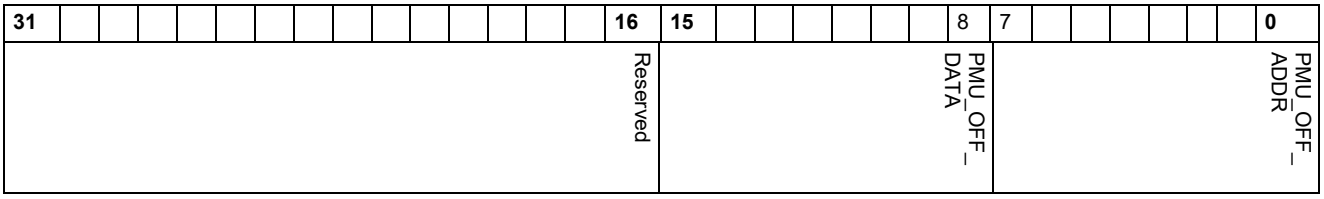
13.1.1 Scratch Registers

The SCRATCH54 and SCRATCH55 registers are used in the TSense reset handler in I2C mode.

Note: *The Reserved bits must be set to 0.*

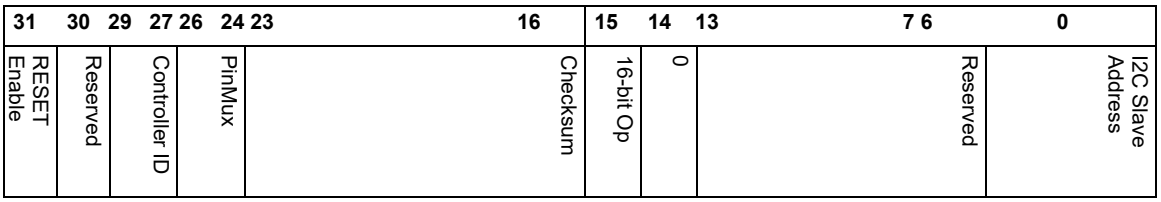
13.1.1.1 SCRATCH54 (PMIC Scratch Register A)

In I2C mode, the SCRATCH54 register bits are configured as shown below.



13.1.1.2 SCRATCH55 (PMIC Scratch Register B)

In I2C mode, the SCRATCH55 register bits are configured as shown below.



13.1.2 Pinmux Support

All Standard I^2C pinmux signals are supported as shown for the three configurations below. Mixed pinmux configurations are not supported; for example, GEN1_I2C_SCL -> I2C1_CLK, SPDIF_IN -> I2C1_DAT is not supported. The I^2C peripheral clock is targeted at 100 kHz (Standard mode). The Controller ID and PinMux columns in the following table are from the SCRATCH55 register.

Table 39: Supported I^2C Pinmux Signals

I2C Controller	Controller ID	I2C Signals	Config 1	Config 2	Config 3
			PinMux = 0	PinMux = 1	PinMux = 2
I2C_1	0	I2C1_CLK	GEN1_I2C_SCL	SPDIF_OUT	SPI2_CS1_N
		I2C1_DAT	GEN1_I2C_SDA	SPDIF_IN	SPI2_CS2_N
I2C_2	1	I2C2_CLK	GEN2_I2C_SCL	-	-
		I2C2_DAT	GEN2_I2C_SDA	-	-

Table 39: Supported I²C Pinmux Signals

I2C Controller	Controller ID	I2C Signals	Config 1	Config 2	Config 3
			PinMux = 0	PinMux = 1	PinMux = 2
I2C_3	2	I2C3_CLK	SDMMC4_CMD	CAM_I2C_SCL	-
		I2C3_DAT	SDMMC4_DAT4	CAM_I2C_SDA	-
I2C_4	3	I2C4_CLK	DDC_SCL	-	-
		I2C4_DAT	DDC_SDA	-	-
I2C_PMU	4	I2CPMU_CLK	PWR_I2C_SCL	-	-
		I2CPMU_DAT	PWR_I2C_SDA	-	-

13.1.3 Checksum Calculation

Every byte in the SCRATCH54 and SCRATCH55 registers must add to 0. The two scratch registers contain 8 bytes total.

To calculate the checksum:

- Write 0 to the 8-bit checksum byte
- Add bytes 0 through 7 where: sum = ~sum+1 (2's complement of sum)
- Write sum to the 8-bit checksum byte.

13.2 Boot ROM Fuses and Straps

IROM Patches, OTP information, and various keys and chip feature configurations are stored in fuses. Refer to Chapter 14: Fuses in this document for more information on fuses.

Straps are used to easily configure certain boot-related information such as boot mode, boot media, and DRAM configuration selection from the BCT.

Table 40: Boot ROM Related Fuses

Number of Bits	Location	Function	Values / Notes
1	FUSE_SECURITY_MODE	OdmMode and JTAG disable	Boolean
128	FUSE_PRIVATE_KEY0 FUSE_PRIVATE_KEY1 FUSE_PRIVATE_KEY2 FUSE_PRIVATE_KEY3	Secure Boot Key (SBK)	128-bit AES key
32	FUSE_PRIVATE_KEY4	Device Key (DK)	32-bit key
14	FUSE_BOOT_DEVICE_INFO [13:0]	Boot Device Config	See below
3	FUSE_RESERVED_SW[2:0]	Boot Device Select	0x0 = eMMC 0x1 = SPI 0x2 = SATA 0x3 = Reserved 0x4 = Reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved
1	FUSE_RESERVED_SW[3]	Skip Device Selection straps	Boolean
8	FUSE_SKU_INFO[7:0]	SKU	Only [3:0] actually used in the Boot ROM

Table 40: Boot ROM Related Fuses

Number of Bits	Location	Function	Values / Notes
4	FUSE_RESERVED_SW[7]	USB Controller select for RCM	[0] = USB 2.0 controller [1] = XUSB controller Note: Either choice is still at USB2.0 speeds.
	FUSE_ENABLE_2_BUTTON_RCM	Forced RCM	If this fuse is burned, both the VOLUME_UP and HOME button have to be pressed during cold-boot starting process to activate Forced RCM. If the fuse is not burned, only the VOLUME_UP button has to be pressed to activate Forced RCM.
	FUSE_RESERVED_SW[5]	ENABLE_WATCHDOG	ENABLE_WATCHDOG = 1 enables watchdog. This fuse should be set to 1 for ODM mode only
	FUSE_RESERVED_SW[4]	SW Reserved	ENABLE_CHARGER_DETECT
256	FUSE_PUBLIC_KEY0	Public key hash	SHA-256 (256 bits) hash of public key modulus
	FUSE_PUBLIC_KEY1		
	FUSE_PUBLIC_KEY2		
	FUSE_PUBLIC_KEY3		
	FUSE_PUBLIC_KEY4		
	FUSE_PUBLIC_KEY5		
	FUSE_PUBLIC_KEY6		
	FUSE_PUBLIC_KEY7		
1	FUSE_PKC_DISABLE	PKC disable control	Not-zero = PKC disable
4	FUSE_SECURE_PROVISION_INDEX	Factory Secure Provisioning Anti-Cloning Key Number	Burned by NVIDIA; Only 15 possible values from index 1 to 15;
2	FUSE_SECURE_PROVISION_INFO	Factory Secure Provisioning control bits	[0] is hide bit; [1] is test_part bit.

Table 41: Fuse Boot Device Information

Device	Fuse Bits	Description	Values
eMMC	13	Reserved	Ignored; set to 0x0
	12:10	MultiPage support	0x0 = default Single page read (512 Bytes)
			0x1 = Multi 2 page read (1024 Bytes)
			0x2 = Multi 4 page read (2048 Bytes)
			0x3 = Multi 8 page read (4096 Bytes)
			0x4 = Multi 16 page read (8192 Bytes)
			0x5 – 0x7 = Reserved
	9:6	Clock Divider PLLP clock @ 408 MHz (Can be SDR or DDR. Select via the DDR Mode Selection bit.)	0x0 = Default clock divider 16 (clock @25.5 MHz)
			0x1 = Clock divider 8 (clock @ 51 MHz)
			0x2 = Clock divider 2.5 (clock @ 163.2 MHz)
			Reserved
			Reserved
			Reserved
			Reserved
			Reserved
			Reserved
			Reserved
	5	VDDIO_SDMMC4 Pads voltage	0x0 = 1.8V (power on default)
			0x1 = 1.2V
	4	Disable Boot Mode	0x0 = Boot mode On
			0x1 = Boot mode Off
	3:2	Voltage Range	0x0 = Query Voltage
			0x1 = High Voltage
			0x2 = Dual Voltage
			0x3 = Low Voltage
	1	DDR Mode Selection	0x0 = Normal
			0x1 = DDR
	0	Data bus width	0x0 = 4 bits
			0x1 = 8 bits
SPI Flash	13:1	Reserved	Ignored; set to 0x0
	0	Page size (Block size is fixed to 32K due to block erase command's limitation)	0x0 = 2K
			0x1= 16K (Boot ROM data buffer size).

Table 41: Fuse Boot Device Information

Device	Fuse Bits	Description	Values
SATA	13:12	Reserved	Ignored; set to 0x0
	11:9	PageSizeMultLog2	Page Size = 512 * (2 ^ PageSizeMultLog2). For PIO mode and DMA to IRAM, Page Sizes less than 32 KB (block size for SATA driver) are supported. For AHCI DMA to DRAM, the page size allowed can be up to the maximum value, 512 * (2^111'b).
	8:6	Wait for COMINIT	If value is N, then wait for COMINIT = 200 * (N+1) ms.
	5	Mode	0x0 = Legacy PIO
			0x1 = AHCI DMA
	4	Force GEN1	0x0 = Default behavior. GEN1 behaves as GEN1, and GEN2 behaves as GEN2
			0x1 = Force GEN2 drive also to behave as GEN1
	3:2	Number of retries for COMRESET	0x0 = Don't retry sending COMRESET
			0x1 = Retry sending COMRESET once
			0x2 = Retry sending COMRESET twice
			0x3 = Retry sending COMRESET three times
	1	Reserved	
	0	PLLE clock source	0x0 = Oscillator if 12MHz, otherwise divided PLLP
			0x1 = Use divided PLLP
USB3/XUSB	13:9	Reserved	
	8	VBUS number	0x0 = VBUS Enable 0
			0x1 = VBUS Enable 1
	7:5	OC Pin	0x0 = OC Detected 0
			0x1 = OC Detected 1
			0x2 = OC Detected 2
			0x3 = OC Detected 3
			0x4 = OC Detected Vbus Pad 0
			0x5 = OC Detected Vbus Pad 1
	4	Reserved	
	3:0	USB port number	0x0 = port0
			0x1 = port1
			0x2 = port2
			0x3 = port3

Table 42: Boot ROM Straps

Number of Bits	Location	Function	Values	Notes
2	RAM_CODE[1:0]	Selects SDRAM configuration set within the BCT	STRAPPING_OPT_A[5:4]	RAM_CODE. SDRAM configuration index selection. {UART1[TXD], UART1[RTS]} = STRAPPING_OPT_A[5:4]
1	RCM_STRAPS	Forces USB Recovery mode or RCM Debug mode	STRAPPING_OPT_A[12:10] + FUSE_ENABLE_2_BUTTON_RCM	{home, vol_down, vol_up} = STRAPPING_OPT_A[12:10]
3	BOOT_SELECT[2:0]	Device Selection	STRAPPING_OPT_A[28:26]	{UART3[TXD], UART4[TXD], UART4[RTS]} = STRAPPING_OPT_A[28:26]
			FUSE_PRODUCTION_MODE = 1	
			0x3 = Use Fuses	The default fuse configuration when no BOOT_DEVICE_INFO fuses are burned is "eMMC x4 BootModeOn"
			FUSE_PRODUCTION_MODE = X (don't care)	
			0x0 = eMMC x8 Boot ModeOff, 512-byte page size, 26 MHz	
			0x1 = SPI	
			0x2 = SATA	
			0x4 = Reserved	
			0x5 = Reserved	
			0x6 = Reserved	
			0x7 = Reserved	

CHAPTER 14: HOST SUBSYSTEM

The host controller (referred to as Host1x) provides a sophisticated programming and control interface to various graphic and video engines, and display. In addition to Host1x's control interfaces with these units, it also has a slave (IP) interface to the ARM7 crossbar (xbar), TSEC, and a direct memory interface to fetch command structures from system memory. Commands are either gathered from a push buffer in memory or provided directly by the CPU, and then supplied to the clients behind Host1x via Host1x channels. The channels also provide a means of synchronization between software and any individual block or amongst the blocks themselves via hardware Sync Point signals (syncpts).

14.1 Glossary

Term	Definition
CDMA	The command DMA. It is responsible for reading commands from memory and supplying them to all channels. It starts from the address in the channel's DMASTART register and continues until it reaches the address in the channel's DMAPUT register.
Channel	A piece of hardware that provides a programming interface to one or more classes. A channel contains a sequence of commands embodied in a command FIFO. This sequence of commands can also be thought of as a thread of execution and of a single context. There exists only one sequence of commands or context per channel. That is to say, there is no hardware-managed context switching within a single channel.
Channel Commands	Entries in the command FIFO. Commands take a common form interpretable by Host and are used for 3 main functions: controlling class ownership and class virtualization of the channel; command expansion via gather commands; and issuing class methods.
Channel switch	A transfer of ownership of a client from one channel to another; a type of context switch. Sometimes a source of confusion, this is not a transfer of ownership of a channel, but of a client. This is the preferred narrowed nomenclature to context switch in order to avoid confusion.
Class	An abstraction of a client, device, or resource. It is a collection of methods. A class can only be owned by multiple channels, but only one channel may actively be using any class, which is known as a channel's working class. The transfer of working class from one channel to another is known as a context switch, which is managed by Host.
Class ID	A unique tag corresponding to each class.
Class method	A method that belongs to an unspecified class.
Class switch	A change in the current in the active class of a client; a type of context switch. In the event that a class switch also involves a channel switch, channel switch is the preferred declaration.
Client, device, resource	A piece of hardware that resides behind Host, connected via the HRD and HRW buses. Client is the preferred terminology for this document. Generally mapped one-to-one with a class, although this is not a strict requirement.
Command FIFO	Holds channel commands supplied by the CDMA or PIO accesses. It is stalled by synchronization methods, back pressure from its destination client, or ownership of the destination client by a different channel
Context switch	A Host event where it facilitates a client's state transition. A context switch is either a channel switch or a class switch.

Term	Definition
Direct register access	Access to a client initiated by the CPU. If a channel and the CPU initiate a data transaction to the same client at the same time, the CPU gets priority. Direct register accesses are orthogonal to channels in regards to client ownership.
DMAGET register	Holds the current address of the CDMA. One exists per channel.
DMAPUT register	Holds the end address of a command stream in memory. One exists per channel.
DMASTART register	Holds the start address of a command stream in memory. One exists per channel.
Gather	A channel command to gather contiguous chunks of memory and inserts it into the command stream.
Host Master	General term to indicate an entity that can control Host. The current list is the CPU, BPMP-Lite, and TSEC.
Host method	A method that belongs to the Host class.
Increment	A channel command that specifies an offset and a count. Increment works like nonincrement, except the offset is incremented by one on each on each write.
Indirect register access	Access to a client initiated by the CPU, but requires two Host register accesses. This is deprecated in the SOC version of the host; it comes from the companion-chip mode where fixed-latency interfaces to the CPU may exist. Like direct register accesses, indirect takes priority over channel transactions. Indirect register accesses must be tied a channel because they require a read return FIFO.
Mask	A channel command that specifies an offset and a mask. A mask command works much like increment and non-increment, except the offsets are calculated by looking at the bits set in the mask. The bit position indicates the relative offset from the specified offset in the command. Mask expects the number of words to follow to be equal to the number of bits set in the mask.
Method	An operation on a class. The most common example is a single register write.
Nonincrement	A channel command that specifies an offset and a count. The offset specifies a method in the current class and the count indicates the number of subsequent words to be written at that offset.
Push buffer	A set of commands residing contiguously in memory. It is a communication method between the processor and Host. Commands are placed at the end of the buffer and a pointer is updated in Host.
SetClass	A channel command that takes a class ID as an argument. SetClass is the sole means to indicate the current virtualization of the channel. It also the sole means that a channel can acquire ownership of a class. Subsequent channel commands are directed towards this class.
Sync Point (Syncpt)	A counter used for synchronization. Every time a specified event occurs, the counter is incremented. A channel can wait until the syncpt attains a specified value, or an interrupt can be generated when a specified value is reached.
Teardown	Can be either a module or channel teardown. Essentially, it means all links and references within Host between the specified module or the specified channel are removed and reset.
Tick Count	Running count of the Host clock after it has been enabled.
WAIT	A host method that takes a single vector argument. It is used in conjunction with syncpt. A WAIT stalls the command FIFO until the supplied vector intersects the RAISE register at all.

14.2 Features

The key features of the Host1x controller are summarized below:

- 14 Channels
- 192 Sync Points
- 64 Syncpt Base Registers
- 32-bit Syncpt Comparison
- 32-bit Timeout register
- Stall and transfer counters
- Unit Activity Monitor (ACTMON): NVENC, VIC, NVDEC, and NVJPG
- Master clients: Crossbar and TSEC
- TZ secure bit In MMIO path
- Master Interface protocol: Native Crossbar
- Client Interface protocol: Hwr/Hrd
- 32-bit Host1x2MC address

14.2.1 Class Based Programming

GPUs support a programming interface that is based on writing to offsets within a “class” that implements a function, rather than to specific register offsets and formats.

By using “class” interface register offsets/formats need not to remain fixed so there is nothing chip specific in the API. This will allow both hardware flexibility and software driver compatibility.

14.2.2 Command Buffer DMA

To maximize the Host1x bandwidth, it is important to burst as many write cycles as possible, which requires the processor to first combine writes. The write buffering typically requires sequential offsets to be written and also may not follow strict programming order. The CPU will first store this buffered data directly to memory and then program the Host1x engine to DMA it.

14.2.3 Multiple Channels

A channel is a thread of execution within Host1x. Similar to a multithreaded CPU, a channel helps defining a context that can be used to allow multiple users of the Host1x and plays a role in context switching.

A channel switch does not always require a client module context switch. There will be a state change within the Host1x, but if channels have non-overlapping usage (e.g., possibly a VIC channel and NVENC encode channel), there may be no need to context switch any Host1x client module.

14.3 Hardware Features

14.3.1 Channels

The Tegra® X1 processor has 14 Host1x channels. Each channel has a set of registers and a command FIFO. Each channel can be associated with one or more Host1x clients. The channel is the primary means of delivering commands to clients, which is described in Section 14.6: Host1x Programming Model.

14.3.1.1 Channel Registers

Channel registers are broken into two functional groups: one grouping is associated with the command FIFO and command delivery to clients; the other grouping is associated with register and memory access.

CDMA Registers

- **HOST1X_CHANNEL_DMASTART_0:** This register triggers a DMA fetch from memory for this channel, if PUT register does not equal the GET register.
- **HOST1X_CHANNEL_DMAPUT_0:** This register triggers a DMA fetch from memory for this channel, if the PUT register does not equal the GET register. This address is relative to the DMASTART base address. It does not support byte writes. All 4-byte data needs to be programmed.
- **HOST1X_CHANNEL_DMAGET_0:** This register tracks the MC offset, which DMA engine has read. It gets incremented as entries are loaded from the channels command buffer into the FIFO. This address is relative to the DMASTART base address.
- **HOST1X_CHANNEL_DMAEND_0:** This is the boundary of illegal addresses (either at the end of the push buffer or at the end of physical memory). This is designed to prevent DMA from prefetching illegal addresses. If DMA reaches this address before seeing a RESTART, it will stop. This would be a software error condition.
- **HOST1X_CHANNEL_DMACTRL_0:** The various fields of DMA control register are described as below:
  - **DMAGETRST**: Reset GET pointer to '0'. Useful for cleaning up crashed channels. DMAGET value is not updated instantly. It takes 4 cycles between programming of reset and valid DMAGET.
  - **DMAGETINIT**: Reset GET pointer to the value of DMAPUT when DMAGETRST is asserted.
  - **DMASTOP**: Stop DMA from fetching on this channel.

Note: *A Command DMA channel needs to be enabled for PIO-gather to work.*

Access Registers

- **HOST1X_CHANNEL_INDOFF_0 and HOST1X_CHANNEL_INDOFF2_0:** The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET value is increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA. The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases.
- For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.
- **HOST1X_CHANNEL_INDCNT_0:** Indirect register access count used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.
- **HOST1X_CHANNEL_INDDATA_0:** This register, when written, writes to the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again. The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.
- **HOST1X_CHANNEL_CMDSWAP_0:** Command swap control. Affects swapping on writes to the PIO region and the frame-buffer buffered memory write region.
- **HOST1X_CHANNEL_FIFOSTAT_0:** CFNUMEMPTY is the number of free slots available in the per-channel command FIFO (needed for PIO or polling for completion of a wait).

14.3.1.2 Channel Command FIFO

The command FIFO is loaded either the CDMA or via PIO access. PIO is mainly used for debug purposes, but it allows a Host1x master to fill the command FIFO with channel commands by writing into “command FIFO” region of the channel's address space. PIO Command FIFO accesses should not be issued while a “Gather” process is busy.

A command FIFO has a set of registers that point to a location in memory where a sequence of channel commands resides. This sequence of commands is generally referred to as a “Push-buffer”.

A Host1x master can write either to DMAPUT or DMASTART register to trigger command fetching by the CDMA. DMASTART indicates where to start fetching, and DMAPUT indicates where to stop. DMAGET is a read-only reference to the present location of the command FIFO; it is incremented when the command is popped from the command FIFO. DMAPUT and DMAGET are relative addresses to DMASTART.

DMAEND provides an upper boundary to prevent the CDMA from fetching illegal addresses; fetching will cease when DMAGET equals DMAEND.

The command FIFO can be halted by writing the DMASTOP field in the Dmactrl register. Dmactrl also provides a mechanism to reset the DMAGET pointer (*Dmagetrst* field) to either 0 or to the value of DMAPUT (Dmainitget field).

14.3.1.3 Channel Commands

Channel commands can be split into two categories:

- **Class commands:** A class command is a mechanism to communicate a class write to a client. The channel must have an active class when processing class commands. The active class is set by a SETCL (SetClass) command.
- **DMA commands:** DMA commands control what is being fetched. They are processed at the top of the command FIFO while class commands are processed at the bottom of the command FIFO. This is because DMA commands change the command stream and must be processed before entering the FIFO.

Offsets in class commands are relative to the active class' base as they are limited to only the methods in the active class.

Channel Commands

- **SETCL (SetClass)** – A channel command that takes a class ID as an argument. SetClass is the sole means to indicate the current virtualization of the channel. It also means that a channel can acquire ownership of a class. Subsequent channel commands are directed towards this class.
- **NONINCR (NonIncrement) –** Takes an offset and count as arguments. There will be <count> data following this class command that will be written to the specified offset. Nonincrement indicates that the offset will not be incremented per data write.
- **INCR (Increment) –** Takes an offset and count as arguments. There will be <count> data following this command that will be written starting at the specified offset. The offset is incremented after each write.
- **MASK (Mask) –** Takes an offset and count as arguments. The number of data to write after the command is equal to the number of set bits in the count. Each data is written to the specified offset plus the next active bit location. For example, a count equal to 0x5 would have 2 data that are written to (offset+0) and (offset+2).
- **IMM (Immediate) –** Takes an offset and 16-bit data as arguments. The 16-bit data is written to the offset.

DMA Commands

- **RESTART (Restart or Jump) –** This command specifies an offset relative to DMASTAT. The next command fetched will be from (DMASTART + offset).
- **GATHER** – Command comprises 2 words and has arguments: offset, insert, type, count, and address. When GATHER is processed, <count> data is fetched from the address and inserted into the command stream. If the <insert> argument is enabled, it will insert either an INCR or NONINCR opcode preceding the fetched data, which is specified by <type>.

DMA commands do not need an active class, but often are processed when an active class exists. In the case of GATHER, if <insert> is enabled, there must be an active class.

14.3.1.4 Channel Control

Channel status and control resides in the synchronous register space of the Host1x address map. These registers are aliased in each channel's space. The details of these registers are as below:

- **CH<0..n>_STATUS:** Status includes client ownership and whether the channel is blocked or not (n is the number of channels – 1).
- **CH_TEARDOWN:** Using this register each channel can be reset, which is referred to as a teardown. This means all channel states are cleared and all client and class ownerships are relinquished.
- **MOD_TEARDOWN:** Similar to channel teardown, there exists a mechanism to reset modules. Setting this register can clear all states associated with a given client/module.
- **<client>_STATUS**: This register indicates the client's currently active class.

14.3.2 Host1x Class

Host1x has its own class that can be executed from the command FIFO. It is comprised of methods to access client registers as well as methods to control channel flow.

14.3.2.1 INDOFF

The following Host1x methods operate identically to the channel registers used for indirect access. Refer to Section 14.8: Host Channel Registers and Section 14.4.7: Indirect Register Access.

- INDOFF
- INDOFF2
- INDCTRL
- INDDATA

INDCNT is absent from the list above. In Host1x master-initiated indirect register reads, INDCNT is written to trigger the read. Conversely, reads from the command FIFO are trigged by a write to INDDATA. Reads issued from the command FIFO are returned to the channel's register return FIFO (return_FIFO).

14.3.2.2 DELAY_USEC

Delay_Usec stalls the command FIFO for the number of microseconds specified. This command has no impact on indirect register accesses if **not** initiated from the command FIFO. The microsecond period is calculated based on setting in the Usec_Clk register. The Usec_Clk register is programmed on the basis of Host1x clock; for example, if the host clock is 250 MHz, then this register should be programmed to a value of 250.

14.3.2.3 TICKCOUNT

Tickcount_Hi, Tickcount_Lo, and Tickctrl can also be controlled from the command FIFO. Their operation is identical whether controlled from the command FIFO or a Host1x master.

14.3.2.4 INCR_SYNCPT

All Host1x modules, including the Host1x itself, implement the Incr_Syncpt class.

Incr_Syncpt <Cond> <Indx>

For the Host1x, this method immediately increments Syncpt[Indx] irrespective of the cond.

14.3.2.5 WAIT_SYNCPT

Wait on syncpt – the command dispatch will stall until the syncpt counter pointed by the index field reaches the threshold value specified in the Thresh field:-

Wait_Syncpt <Indx> <Thresh>

The channel will wait until the following is true:

Syncpt[Indx][15:0] >= Thresh[15:0]

where the “>=” takes into account wrapping (see “Section 14.3.5: Sync Points (SYNCPTs)” for more information on wrapping). More specifically, the channel will wait until:

((Syncpt[indx] – thresh) & (1 << 15))!= 0

14.3.2.6 WAIT_SYNCPT_BASE

This method uses Syncpt Base registers to calculate threshold value for channel wait operation. The syncpt index, base index and also optional offset will be a part of Wait_Syncpt_Base command:-

Wait_Syncpt_Base <Indx> <Base_Indx> <Offset>

The channel will wait until the following is true:

Syncpt[Indx][15:0] >= (Syncpt_Base[Base_Indx] + Offset)[15:0]

Where the “>=” takes into account wrapping. See Section 14.3.5: Sync Points (SYNCPTs) for more information on wrapping.

14.3.2.7 WAIT_SYNCPT_INCR

Wait until syncpt increments:

Wait_Syncpt_Incr <Indx>

The channel will stall until Syncpt[Indx] is incremented. This wait method is not recommended.

14.3.2.8 LOAD_SYNCPT_BASE

Load a new value into the Syncpt_Base register:

load_syncpt_base <base_indx> <value>

Syncpt_Base[Base_Indx] will be loaded with <Value>.

14.3.2.9 INCR_SYNCPT_BASE

Add an offset to the value in the Syncpt_Base register:

Incr_Syncpt_Base <Base_Indx> <Offset>

The following operation will be done:

Syncpt_Base[Base_Indx] += Offset

14.3.2.10 STALLCOUNT

STALLCOUNT_HI, STALLCOUNT_LO, and STALLCTRL are methods to control “stall counters” through the command FIFO. Details of stall counters are given later in this document.

14.3.2.11 XFERCOUNT

Channel_Xfer_Hi, Channel_Xfer_Lo, and Xferctrl are methods to control “xfer counters” through the command FIFO.

14.3.2.12 32-Bit Sync Point Comparison Methods

Five methods are used for 32-bit sync point comparison. Software should send these commands using the following format:

LOAD_SYNCPT_PAYLOAD_32, <other_command>

where <other_command> is one of the following::

- WAIT_SYNCPT_32
- WAIT_SYNCPT_BASE_32,
- LOAD_SYNCPT_BASE_32,

- INCR_SYNCPT_BASE_32

LOAD_SYNCPT_PAYLOAD_32 <Payload(32)>

This method loads a 32-bit value into the corresponding channel's Channel_Syncpt_Payload register:

Channel_Syncpt_Payload[31:0]= <Payload(32)>

WAIT_SYNCPT_32 <Indx(8)>

This method stalls the current channel until following condition is true:-

(SYNCPT [<indx>][31:0] - PAYLOAD[31:0]) & 0x80000000) == 0

Here the Payload value is taken from Channel_Syncpt_Payload of the current channel.

This is essentially a wrapping stall until ((SYNCPT [<indx>][31:0] >= PAYLOAD[31:0])

WAIT_SYNCPT_BASE_32 <Indx(8)> <Base_Indx(8)>:

This method stalls the current channel until following condition is true:-

((Syncpt [<Indx>][31:0] - (Payload[31:0] + Syncpt_Base[<Base_Indx>][31:0]) & 0x80000000) == 0

Here the Payload value is taken from Channel_Syncpt_Payload register of the current channel.

This is essentially a wrapping:-

stall until ((Syncpt [<Indx>][31:0] >= (Payload[31:0] + Syncpt_Base[<Base_Indx>][31:0]))

LOAD_SYNCPT_BASE_32 <Base_Indx(8)>

This method copies the value from the current channel's Channel_Syncpt_Payload register into the Syncpt_Base register specified through the Base_Indx field:-

Syncpt_Base[<Base_Indx>][31:0] = Channel_Syncpt_Payload[31:0]

INCR_SYNCPT_BASE_32 <Base_Indx(8)>

This method adds the value of the current channel's Channel_Syncpt_Payload register into the Syncpt_Base register specified through Base_Indx:-

Syncpt_Base[<Base_Indx>][31:0] += Channel_Syncpt_Payload[31:0]

14.3.3 Context Switching

Context management of Host1x 1x-based modules will be completely under software control (there will be no automatic context switching done by hardware). This mean a module will never produce a context switch interrupt, it will always operate in auto-acknowledge mode.

14.3.4 Behavior of SetClass

The SetClass does not mean implicit acquisition of a module’s ownership. It is possible to send commands simultaneously from more than one channel to the same module. If exclusive ownership of a module is required for software correctness, then acquire_mlock and release_mlock opcodes should be used, which acquire and release the MLOCKn semaphore bits. An acquire_mlock that fails will cause that channel's commands to stall until it wins subsequent MLOCK arbitration (arbitrations happen with each release_mlock).

Here is the summary of SetClass behavior:

- SetClass simply instructs the channel hardware which module and context to use for the following commands.
- If multiple channels write simultaneously to the same class ID, and no MLOCKs are used, then the actual commands are interleaved in round-robin fashion (one command per channel).

- If multiple channels write simultaneously to different class IDs in the same hardware module (and no MLOCKs are used), then the actual commands are interleaved, but in blocks of N-cycle bursts (each burst of commands must be preceded with a CTXSW to the module giving the new context ID).

  N in this case is programmable number configured through Ctxsw_Timeout_Cfg register.

14.3.5 Sync Points (SYNCPTs)

A Sync point is a mechanism to synchronize between software and Host1x clients and also in between Host1x clients. This is implemented as 32-bit counters which are incremented by 1 whenever some predestinated condition (or event) occurs. When a counter reaches its maximum value, it wraps back to zero on the next increment.

Tegra X1 processors have 192 sync points. Sync points are not permanently associated with a channel; sync point allocation is done by software during initialization time.

There are two basic ways for sync point increments: -

- When a CPU writes an index to the Host1x's "syncpt_cpu_incr" register.
- When a Host1x client has received an "incr_syncpt" method and the condition specified by this method has become true.

Synchronization using sync points can be done in the following ways:

- A CPU can be interrupted when a sync point reaches a pre-specified value.
- A Host1x channel can have "wait" commands so that channel will wait for a pre-specified sync point value.

Sync points are normally not reset, but can wrap -- the comparison takes into account the possibility of wrapping.

Note: *Sync point wrapping works only if (Syncpt value - Syncpt Threshold) <= $2^{(syncpt\_width-1)}$. For 16-bit syncpt comparisons, the difference should be less than or equal to 32768. Software must take care of wrapping issues. For 32-bit comparisons, the difference should be less than or equal to 2,147,483,648, Because of the large value, software will not see any wrapping issues.*

All Host1x clients (e.g., VI) implement the following increment syncpt method:

Incr_Syncpt <Condition> <Index>

The Host1x client would receive the "Incr_Syncpt" method and store the index for each condition. Whenever the "condition" event occurs, the client would return the index back to Host1x.

14.3.5.1 Client Model for Syncpt Behavior

Software Programming Model

The basic programming model that software will follow is:

Each module will be programmed to do a unit of work (an operation) by Host1x using CDMA and push buffers.

Examples of an operation include:

- BLT (VIC)
- Draw a set of triangles (GPU)
- Encode a single frame (NVENC).

If nothing else is programmed, then module will go idle until Host1x sends commands to start another operation (no continuous mode).To do its operation, a module reads data from memory and writes the results to memory. Modules interact with each other using memory buffers: one module is the producer of data, and another is the consumer of that data.

There are exceptions to this model which we will discuss in detail later, but should be mentioned here. The exceptions include VI and DISPLAY which work in continuous mode (they continue to process data without the need of additional Host1x programming).

Basic Synchronization

There are two basic needs for synchronization: management of memory buffers and timing of control register writes.

Memory buffers used to pass data from one module to the next use a producer/consumer model with circular buffers. To prevent buffer underflow and overflow, synchronization needs to be done in both directions:

- The consumer cannot read until the producer is done writing (and the writes are committed to memory).
- The producer cannot reuse an output buffer (i.e., write to buffer) until the consumer is done reading the buffer.

Thus, the synchronization events required for memory buffers are:

- The module has completed all reads from the buffer.
- The module has completed all writes to the buffer (and they are committed by the memory controller).

To understand the requirements for timing the writes to control registers, a typical sequence is provided below:

1. reg wr for operation A
2. reg wr for operation A
3. reg wr (trigger) for operation A
4. reg wr for operation B
5. reg wr for operation B
6. reg wr (trigger) for operation B

If no WAIT method is placed between the trigger for A and the first register write for B, then in the worst case, corruption of operation A may occur because the new value of the control register is used before operation A is done. For modules that protect against this corruption, there is still the undesirable behavior of the module delaying the register write and subsequently causing back pressure on the Host1x write bus. If we wish to allow direct register reads to be used by ISRs they will happen asynchronously to the channel command writes, so one must ensure that the Host1x bus does not stall for significant periods of time.

For synchronizing writes to control register, a safe time to start writing register for the next operation is defined to be when:

- No corruption will occur for previous operations.
- No stall of the HWRBUS bus will result.

14.3.5.2 Standard Set of Incr_Syncpt Conditions

The following values of the “incr_syncpt cond” field are predefined:

- **0** (Immediate): Return indx to Host1x immediately (used by software push-buffer allocation and helpful for debug).
- **1** (Op_Done): Return indx to Host1x when all previously triggered operations have completed and their writes to memory are committed.
- **2** (Rd_Done): Return indx to Host1x when all previously triggered operations have completed their reads from memory.
- **3** (Reg_Wr_Safe): Return indx to Host1x when it is safe to program registers for the next operation. Safe means no corruption to previous operations and no stalling of Host1x write bus will occur.

There are special cases which would require use of additional condition values. Some modules have multiple read buffers condition = 2 (all reads done) would mean all reads to all buffers done, but it might be useful to software to know when reads are done to a specific input buffer. For some modules, e.g., VI, there are different safe times to update different sets of registers, so condition = 3 will need several variations (one for each "safe time"). If a module can have two operations happening at one time

(as in VI), then special considerations will need to be made for these cases: either two separate Incr_Syncpt methods or one Incr_Syncpt method with many special conditions.

For some modules, one condition can replace another if the two conditions always happen within a short period of time of each other. Then the condition that always happens last can be used for both.

14.3.5.3 Continuous Mode — Display and VI

For each display, software wants a free-running syncpt increment on every display “Vsync”. The most appropriate implementation of this is to have an additional register which has “Enable” and “Indx” fields to control the increment of syncpt on every “Vsync” event. If enable is set, then on every “Vsync”, display would return this “Indx” back to Host1x.

A similar situation exists for VI, where besides a set of conditions for the “Incr_Syncpt” method, it would also need a register with “Enable” and “Indx” which would control the incrementing of a syncpt on every camera “Vsync”.

Host1x clients will implement the logic associated with these registers. When this continuous mode is enabled, the client should set a pending bit for when the condition occurs. When the pending bit is set, the client will arbitrate for the hrd_<client>2host1x bus and if selected it will send “Indx” tagged with “Syncpt type” on hrd_ bus. After successfully sending the indx on hrd bus, the client will clear the pending bit.

One of the issues for continuous operation of VI is that not all modes of operation allow the consumer of their output buffers to signal back pressure when the consumer has fallen behind in reading. To alleviate this problem, VI has registers which, when written to, signal that the reading of one buffer has completed.

See Table 44 for a list of the Host1x clients.

14.3.5.4 Allowing Multiple Pending INCR_SYNCPTS

Clients can have multiple pending syncpts which are queued into the client’s syncpt FIFOs, which are dedicated ones for each syncpt conditions.

The behavior of these syncpt FIFOs is controlled by an “Incr_Syncpt_Cntrl” register in client’s register space:

INCR_SYNCPT_CNTRL< No_Stall>< Soft_Reset>

- No_Stall:

  If this bit is 1 and an Incr_Syncpt method is received but the syncpt FIFO for that condition is full, then this method will be dropped and the Incr_Syncpt_Error[Cond] bit will be set.

  If this bit is 0, then instead of dropping next method in case of a syncpt FIFO full condition, the client will just stall the host interface.

- Soft_Reset:

  if Soft_Reset is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set Soft_Reset of all Host1x clients affected, then clear all Soft_Resets.

INCR_SYNCPT_ERROR<Cond_Status>

This register stores error status in case of above mentioned syncpt FIFO full condition.

14.3.5.5 32-bit Sync Point Comparison

In Tegra 3 and previous chips, sync point counters were implemented as 32-bit registers but comparison takes into account 16-bit value only. In Tegra X1 devices, sync point comparison is extended to 32 bits. To support this feature, Host1x has the following changes:-

Host1x 32-Bit Registers

The following registers are extended to 32 bits:

- Syncpoint base register [Host1x 1x_Sync_Syncpt_Base]

- Syncpt Threshold register [Host1x 1x_Sync_Syncpt_Int_Thresh]

The following 32-bit register stores the payload value required in syncpt methods:

- Channel_Syncpt_Payload[31:0]

Host1x Method Changes

To support 32-bit syncpt comparisons, new Host1x methods have been introduced and are described later in this document.

14.4 Unit Description

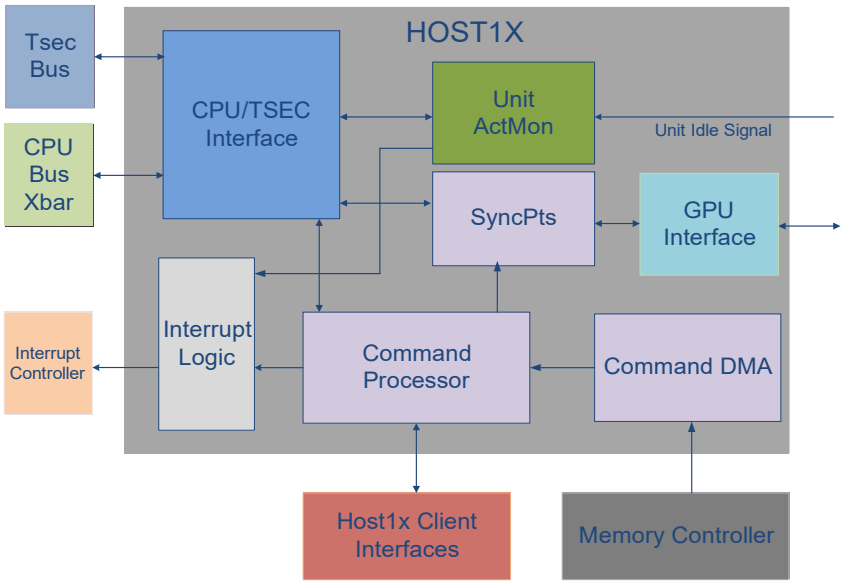
Host1x has a master client interface with the CPU Xbar bus and also a TSEC unit to receive CPU/TSEC read/write commands. The incoming commands are either processed inside Host1x if they are Host1x specific or routed to clients.

Host1x has a point-to-point interface with its clients using HWR/HRD buses, which are used for sending requests and accepting responses from clients.

There is a memory controller interface to fetch Push-Buffer commands from memory through the “Command DMA” unit. The incoming Push-Buffer commands are processed inside “Command Processor” and afterwards either consumed inside Host1x or it will generate tractions to client interface. There is an internal “syncpt” unit which is used to synchronize between Host1x clients and also between software.

There is a “Unit ActMon” block to monitor activities of Host1x clients. The ActMon statistics are used by software for power management.

Figure 23: Host1x Top-Level Block Diagram

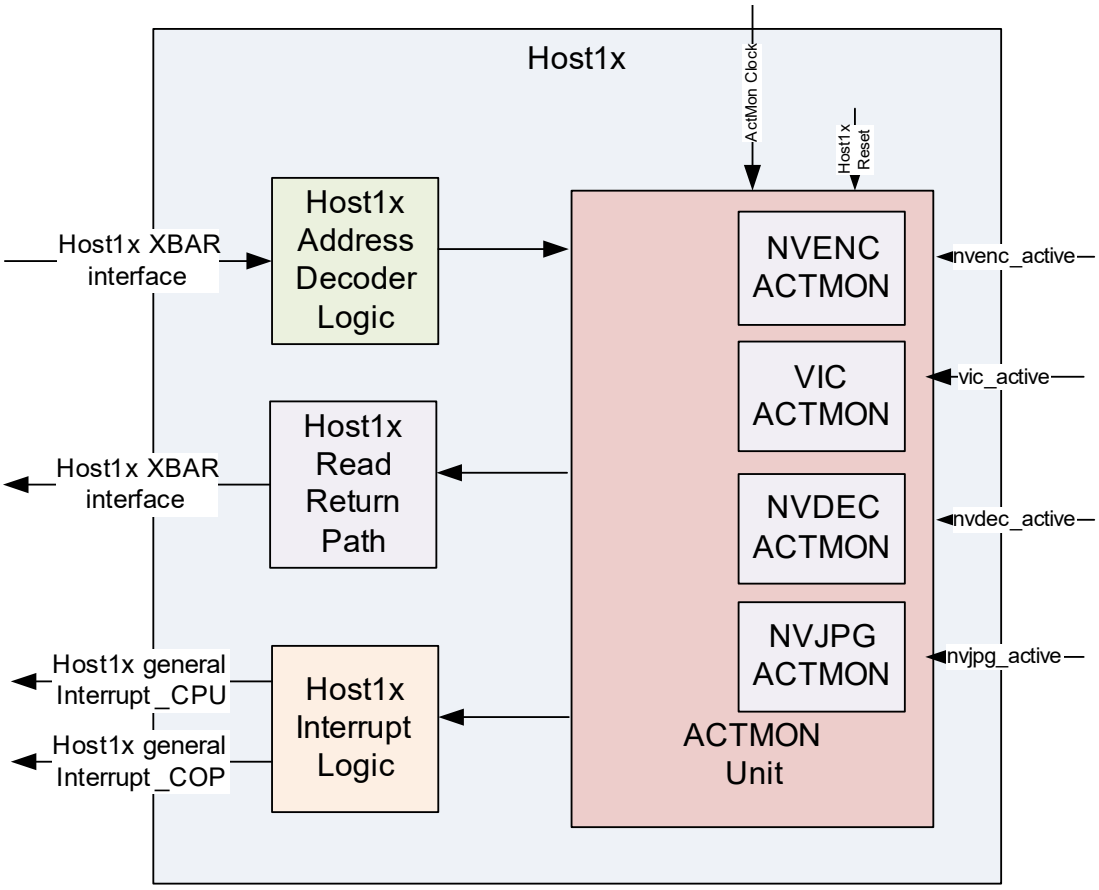


14.4.1 Activity Monitor

The Host1x unit implements 32-bit activity monitor (ACTMON) counters to monitor the idleness of the NVENC/VIC units.

The NVENC/VIC units send active information to Host1x through the same “active” signal (which is a level signal). The ACTMON block provides averaging, watermark detection, and interrupt functionality similar to the central activity monitors. The Host1x driver manages these monitors.

Figure 24: ACTMON Interface



The clock for the ACTMON unit is a low-frequency (10 MHz ~ 50 MHz), non gateable clock.

There is 1 interrupt status bit (per NVENC/VIC/NVDEC/NVJPG ACTMON interrupt) in Host1x, which is set by the actmon_intr_set_msenc/nvenc/nvdec/nvjpg/vic signals and cleared by software. Also there are corresponding interrupt mask bits for these interrupt lines to enable/disable interrupts.

The ACTMON register accesses are handled by the ACTMON itself. The Host1x will forward the access control/address/data signal to the ACTMON unit.

The ACTMON registers are a part of Host1x register space.

14.4.2 Filtering Out Kernel Commands from the Gather Buffer

This feature maintains user mode driver versus kernel driver protection. The host1x kernel driver sets up mode bits, which need to be protected from the user mode driver, so that driver cannot directly break the kernel.

Host1x checks the incoming commands from the gather buffer. If it contains the SetClass kernel mode command, then Host1x should drop these commands and raise an “Invalid Gbuffer cmd” interrupt to software. Software will read the interrupt status register to detect the channel that received the invalid gbuffer command. That particular channel needs to be restarted through the “channel teardown” mechanism.

- HOST1X_SYNC_HINTSTATUS_EXT1_0[0:11]

  Bit (i):-ch(i)_Invalid_gbuffer_cmd_int_status

  Where i = 0..11
- HOST1X_SYNC_HINTMASK_EXT1_0[0:11]

  Bit (i):- ch(i)_Invalid_gbuffer_cmd _int_mask

  Where i = 0..11

There might be some cases where a channel can still see setClass, for example, context switching. For those channels this feature must be disabled. This feature is controlled through CHANNELCTRL for each channel:

CHANNELCTRL[2]

- 0: disable filtering of kernel command(setClass).
- 1: enable filtering of kernel command(setClass).

14.4.3 Timeout Mechanism

Host1x has a mechanism to generate an interrupt for unserviced CPU read/write requests, which is controlled through a timeout register (HOST1X 1x_IP_TIMEOUT). Host1x generates a timeout Interrupt, and a violating address is stored in the following registers:

- IP_READ_TIMEOUT_ADDR
- IP_WRITE_TIMEOUT_ADDR

The width of the timeout register is 32 bits to provide a timeout value of ~19 seconds.

The TSEC master client interface also uses this mechanism to generate a timeout interrupt to CPU in case of illegal read/write access. The HINTSTATUS_EXT register has been modified to indicate if the timeout is from native CPU access or TSEC access.

Also in case of a TSEC read timeout, the returned response packet will have an additional bit (tsec2host1x 1x_iprdata_timeout) to indicate a timeout error.

HINTSTATUS[Timeout_source=bit10] =

- 0 Timeout from TSEC access.
- 1 Timeout from XBAR access.

The current scheme saves only the last timeout address in case of multiple timeout happens before clearing the timeout interrupt by software.

14.4.4 Performance Statistics Counters

Host1x has performance counters for profiling, which can be enabled independent of push-buffer commands. Host1x needs the following statistics per channel:

- Total clocks
- Clocks stalled waiting for syncpt
- Clocks transferring data

14.4.4.1 Tick Counter

Each channel has its own 64-bit tick counter that is incremented on each Host1x clock, given that it is enabled. TICKCTRL enables and disables the tick counter.

The 32-bit TICKCOUNT_LO and TICKCOUNT_HI registers can be written to initialize the tick counter. Reads of these registers return the tick count. TICKCOUNT_LO stores the lower 32 bits, and TICKCOUNT_HI stores the upper 32 bits. TICKCOUNT_HI is calculated as follows:

```
if (TICKCOUNT_LO == 0xFFFF_FFFF && TICKCTRL == enable) TICKCOUNT_HI++;
```

14.4.4.2 Channel Stall Counter

The 64-bit CHANNEL_STALL counter per channel, when enabled, is incremented on each clock cycle if that particular channel is waiting for syncpt.

The following registers control this counter:

- STALLCTRL: This 1-bit register enables/disables the CHANNEL_STALL counter.
- STALLCOUNT_LO: This 32-bit register initializes the lower 32 bits of the CHANNEL_STALL counter.
- STALLCOUNT_HI: This 32-bit register initializes the upper 32 bits of the CHANNEL_STALL counter.

The following Host1x class methods are used to program this counter:

- STALLCOUNT_HI: This method initializes the high 32 bits of the tick count value in the CHANNEL_STALL counter.
- STALLCOUNT_LO: This method initializes the low 32 bits of the tick count value in the CHANNEL_STALL counter.
- STALLCTRL: This method enables/disables the CHANNEL_STALL counter.

14.4.4.3 Channel Xfer Counter

There is a 64-bit CHANNEL_XFER counter per channel to measure the data transfer interval per channel. This counter is incremented at each clock cycle, if the channel is not idle (busy in fetching channel commands).

The following registers are used to initialize this counter:

- XFERCTRL: This 1-bit register enables/disables the CHANNEL_XFER counter.
- CHANNEL_XFER_LO[31:0]: This 32-bit register initializes the lower 32 bits of the CHANNEL_XFER counter.
- CHANNEL_XFER_HI[31:0]: This 32-bit register initializes the lower 32 bits of the CHANNEL_XFER counter.

The following Host1x class methods are used to program this counter:

- CHANNEL_XFER_HI: This method initializes the high 32 bits of the tick count value in the CHANNEL_XFER counter.
- CHANNEL_XFER_LO: This method initializes the low 32 bits of the tick count value in the CHANNEL_XFER counter.
- XFERCTRL: This method enables/disables the CHANNEL_XFER counter.

Note: *Counter operations are identical irrespective of whether they are controlled through registers or other methods.*

14.4.5 TZ Non-Secure Bit for the Display Engine

To support programming of a Host1x client (display engines) in TZ (Trust Zone) mode through direct register access, the Host1x needs to transfer a TZ non-secure bit from the CPU to the display and VI engines.

The following table describes the behavior of TZ non-secure signal to clients.

Table 43: Behavior of TZ Non-Secure Signal

Access-Mode	CPU TZ NS bit	TZ Non-Secure Bit to Client
Direct MMIO access (read/write)	0	0
Direct MMIO access (read/write)	1	1
Indirect MMIO access (read/write)[a]	0	X
Indirect MMIO access (read/write)	1	1
Push-buffer access (write) (DMA/PIO)	X	1
Push-buffer access (Indirect reads) (DMA/PIO)	X	1

a. This mode is not supported for “secure access”, The CPU has to use only “direct access” in “secure mode”.

14.4.6 Interrupts

Host1x has following types of interrupts:

14.4.6.1 Host1x Client Interrupt

Host1x is the collection point for all of its clients interrupts. Also it has control over those using status and mask registers with fields for each client. These client interrupts are forwarded to the central interrupt controller unit, individually through per client interrupt lines.

14.4.6.2 Host1x Interrupts

Host1x -specific interrupts are specified through the HOST1X_SYNC_HINTSTATUS_0 register while additional interrupts have been specified through the HOST1X_SYNC_HINTSTATUS_EXT_0 register (see Registers below). Each Host1x interrupt also has a corresponding mask. An interrupt is cleared by writing a 1 to the corresponding bit.

14.4.6.3 Host1x Syncpt Interrupts

Host1x can generate a syncpt interrupt upon reaching a threshold value by syncpt registers. It can also generate an interrupt to either CPU (referred to as CPU0 below) or BPMP-Lite (referred to as CPU1 below).

An interrupt is routed to cpuN when:

> (SYNCPT[indx] >= SYNCPT_INT_THRESH[indx]) && (SYNCPT_THRESH_INT_MASK[indx] == cpuN)
>
> The comparison takes wrapping into account. See Section 14.3.5: Sync Points (SYNCPTs)for more information on wrapping.

The status is sticky and is PENDING until cleared. Write 1's to clear.

14.4.7 Indirect Register Access

Host1x provides indirect register access to all its clients. An indirect register access involves multiple steps and requires a channel for the read return data. All registers required for indirect accesses reside in a channel.

Writes involve two Host1x register writes in the owning channel's space; the first write indicates the address (INDOFF, or INDOFF2 and INDCTRL), and the second indicates the data (INDDATA).

Reads involve two direct Host1x writes, the first indicates the address (again INDOFF, or INDOFF2 and INDCTRL), and the second indicates the number of reads (INDCNT). It also requires an indeterminable number of direct Host1x reads to the read return FIFO status register followed by a read to the read return FIFO.

- INDOFF
- INDOFF2
- INDCTRL
- INDDATA
- INDCNT

Pitfalls — Indirect accesses require a software mutex to ensure that no other software threads access the indirect registers while issuing the access (since the series of accesses is non-atomic). Indirect reads can only issue counts less than the read return FIFO size to prevent a deadlock – too many read requests can block popping of the read return FIFO. This restriction may be even tighter. All module accesses must also be tied to a channel.

In case an indirect write is sent, software must check whether the previous indirect write is through using an immediate sync increment command to the client.

14.4.8 Direct Register Access

All clients under Host1x have 256KB of address space, which originates from the space specified by indirect offset register (INDOFF). Host1x's client address map is dictated by its client's module IDs; these IDs are used in INDOFF to specify the target module of the register access. A client's address in the Host1x space is calculated as:

> address = direct_access_base_address (0x54000000) + (module_id << 18) + 4 * register_offset

Only one pending read per interface can exist. Reads are returned from the client when the client is ready. There are no latency restrictions.

Indirect versus Direct — a pitfall of direct addressing is variable latency of Host1x's clients. While this has no impact on writes, the CPU is blocked until the read returns rendering the CPU idle during that time.

14.4.9 Host Clients

The following table lists the clients that are accessed via Host1x with their module (client) ID, which is used to determine addresses for direct addressing.

Table 44: Host1x Clients

Client	ID
Host1x	0x0
DPAUX1	0x1
VI	0x2
TSEC1	0x4
Display A	0x8
Display B	0x9
SOR1	0xa
DSI	0xc
VIC	0xd
NVJPG	0xe
CSI	0xf
DSIB	0x10
NVDEC	0x12
NVENC	0x13
TSEC	0x14
SOR	0x15
DPAUX	0x17
ISP	0x18
ISPB	0x1a
VII2C	0x1b

14.4.10 Class IDs

The following table lists the class IDs.

Table 45: Host1x Class IDs

Class	ID
NV_HOST1X_CLASS_ID	0x01
NV_VIDEO_ENCODE_NVENC_CLASS_ID	0x21
NV_VIDEO_STREAMING_VI_CLASS_ID	0x30
NV_VIDEO_STREAMING_ISP_CLASS_ID	0x32
NV_VIDEO_STREAMING_ISPB_CLASS_ID	0x34
NV_GRAPHICS_VIC_CLASS_ID	0x5D
NV_DISPLAY_CLASS_ID	0x70
NV_DISPLAYB_CLASS_ID	0x71
NV_HDMI_CLASS_ID	0x77
NV_DISPLAY_DSI_CLASS_ID	0x79
NV_DISPLAY_DSIB_CLASS_ID	0x7A

Table 45: Host1x Class IDs

Class	ID
NV_SOR_CLASS_ID	0x7B
NV_DPAUX_CLASS_ID	0x7D
NV_NVJPG_CLASS_ID	0xC0
NV_TSEC_CLASS_ID	0xE0
NV_NVDEC_CLASS_ID	0xF0

14.4.11 Host Address Space

Each channel is allocated 16KB of space, and they are aligned contiguously at the top of the Host's address space. A channel's space consists of channel registers, the command FIFO, and an aliased frame buffer region, which are unique per channel. The rest of the space consists of asynchronous, synchronous and read DMA registers, which are aliased in each channel – there exists only one copy.

14.4.11.1 Channel Map

The following table gives offsets from the channel base address for different sets of registers available to each channel.

Table 46: Host1x Channel Map

0x0000	Channel Registers	
0x0600	RDMA Registers	
0x0800	Command FIFO	
0x1000	Frame Buffer	
0x2000-0x20FF	Unit ACTMON Registers	ACTMON1(NVENC) = 0x2000 – 0x203F
		ACTMON2 (VIC) =0x2040 – 0x207F
		ACTMONRSVD =0x2080 – 0x20FF
0x2100*	Synchronous Registers	
0x3FFF	Bottom	

* 0x2100 is the synchronous registers base (HOST1X_CHANNEL_SYNC_REG_BASE)

14.4.11.2 Host1x Map

The following table shows the Host1x addresses for the CPU. Channels start at the lowest addresses and are aligned contiguously. Direct access to Host clients starts at 5400_0000, but this is configurable through a BAR register.

Table 47: Host1x Map

Address Range	Channel
5000_0000-5000_3FFF	Channel 0 (16KB)
5000_4000-5000_7FFF	Channel 1 (16KB)
5000_8000-5000_BFFF	Channel 2 (16KB)
5000_E000-5001_2FFF	Channel 3 (16KB)
5001_3000-5001_6FFF	Channel 4 (16KB)
5001_7000-5001_AFFF	Channel 5 (16KB)
5001_B000-5001_BFFF	Channel 6 (16KB)
5001_C000-5001_FFFF	Channel 7 (16KB)
5002_0000-5002_3FFF	Channel 8 (16KB)
5002_4000-5002_7FFF	Channel 9 (16KB)
5002_8000-5002_BFFF	Channel 10 (16KB)
5002_C000-5002_FFFF	Channel 11 (16KB)

14.5 Performance

14.5.1 Key Use Cases

Command throughput must be sufficient when all channels are active, the CDMA is active, and spooling is active.

14.5.2 Latency Targets

Direct read latency must be under 1 ms. Clients that cannot guarantee that a read will return in under 1 ms must not be read directly but indirectly.

14.5.3 Bandwidth Targets

As Host1x unit is used only for sending control commands to clients and usually at the frame interval.

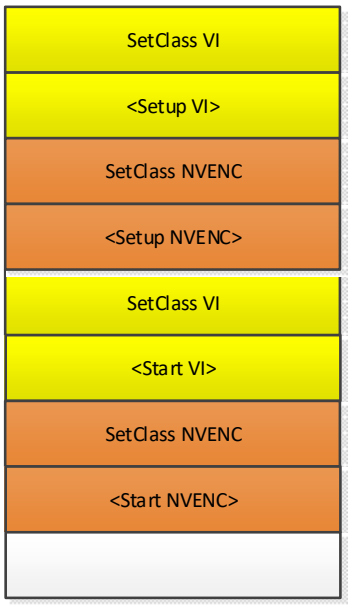
As these control commands are short (~200 to 300 words maximum), it is expected that the bandwidth requirement on each client should be less than 1MB/s, whereas the maximum available bandwidth per client on the HWR bus will be (206*4B)/12 =68MB/s at 206 MHz Host1x clock. The minimum bandwidth on the HWR bus per client is 33.34 MB/s at 100 MHz clock which should easily meet the client requirement.

14.6 Host1x Programming Model

The Host programming models resides on top of the concept of channels. A channel provides the means for software to supply an ordered sequence of commands to one or more classes. There are no restrictions on how many classes a channel may own or how often it can switch between classes, but it can only operate on one class at a time (known as the working class) and one command at a time. A stalled channel does not allow any commands to proceed from any class – there is a strict ordering of commands. A channel starts processing commands simply when commands are present, either supplied by the CPU or gathered from memory. A channel can be stopped explicitly by a CPU write to channel state.

A single channel owning multiple classes and clients:

Figure 25: Single Channel Example



14.6.1 Concurrency

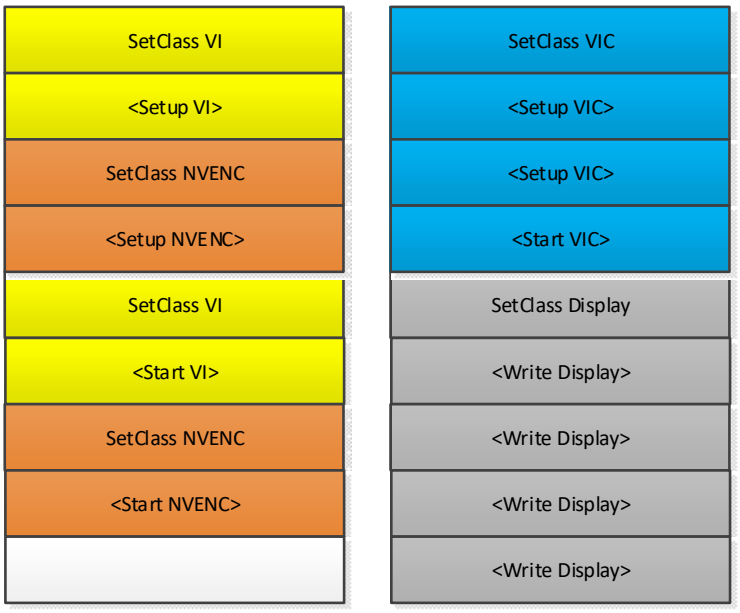
Channels operate in parallel and are unhindered by one another except in two specific cases:

- Synchronization points

- Class contention

The following figure depicts such concurrency.

Figure 26: Channel Concurrency Example



In the example above, no channel is blocked by another at any time so both can work concurrently. Care must be taken when assigning clients to channels. Channel concurrency as well as channel throughput is crucial for performance. For example, a client that is event-driven with a low latency requirement should exist in its own channel. Conversely, two clients that must be steadily supplied with commands could possibly coexist in the same channel given that their synchronization points are not conflicting (if they have any real synchronization points at all). Clients that operate sequentially on the same piece of data can easily reside in the same channel.

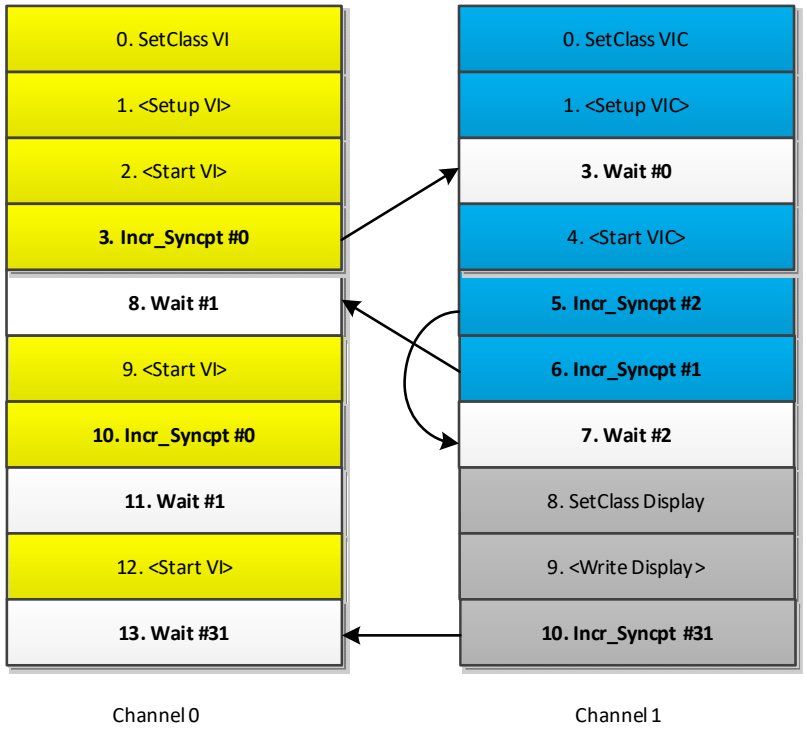
Clients with low-latency requirements like described in the first example above should possibly be moved to a PIO programming model and away from a channel-based model.

14.6.2 Synchronization

There are 192 total synchronization points (syncpts). Similar to semaphores, syncpt methods are issued to clients, mentioning syncpt counter index and returning condition. Any channel can be made to wait on a syncpt value of a particular syncpt index through Host1x “wait” method. Once the syncpt method is received by clients, based on syncpt condition, they will return a syncpt index back to Host1x. On receiving a syncpt index from the client, Host1x will increment that particular syncpt counter.

The following simple example below details how syncpts and waits allow for synchronization. The arrows simply indicate channel dependencies, which channel is waiting for syncpt increment from other channel.

Figure 27: Channel Synchronization Example



14.6.3 Progress Status

Channel progress is monitored through two means – GET and syncpts. GET is a channel state that indicates the address of the last command that has executed in the channel. GET is the same as DMAGET.

Syncpts are counter registers that are incremented when specified events occur (e.g., after the completion of each operation done by a module). The 32-bit syncpt values typically are monotonically increasing and can be used for in channel waits and out-of-channel interrupts.

The GET and syncpt registers can also be read directly (out-of-band) by the CPU.

14.6.4 Syncpt Base Register Use Case

The syncpt base registers are modified through LOAD_SYNCPT_BASE_32 or INCR_SYNCPT_BASE_32 methods and are used with the WAIT_SYNCPT_BASE_32 method.

This mechanism is useful when software wants to wait for a particular syncpt increment, but software does not know the absolute value of the sync point register until later.

The "future-value" of a sync point is the value that the syncpt register contains right after a particular increment has occurred.

Consider this example:

In this example, time increases downward (later (lower) lines are ahead in time).

Software "knows" the syncpointA future-value equals N at this point; software "knows" the baseA future-value equals N at this point.

1. command1
2. increment syncpointA to N+1
3. command2
4. increment syncpointA to N+2
5. command3

6. wait until command1 is done (syncpoint == N+1)
7. command4
8. wait until command2 is done (syncpoint == N+2)
9. command5

If software actually DOES know the value of N then regular WAIT_SYNCPT works fine here. However, the user space software driver (which is creating this list of commands) does not know what the value N is until after these commands (1-9) are all flushed to the kernel space driver.

The kernel space driver receives packets of commands from many different user space processes. When it receives a packet of commands it queues those commands (i.e., writes the Host1x PUT pointer). It also keeps track of all increments that have occurred in all commands that have been queued so far. The "future-value" of each sync point is simply the total number of increments that have occurred since the beginning of time (actually "beginning of time" is really the point in time where the kernel space driver was initialized and reset all the syncpt registers to 0).

So when the kernel space driver queues these commands (1-9), it knows the value N (i.e., the sum of all increments since the beginning of time). It can pass this back to the user space driver. But the user space driver does not know how many increments will be queued before commands 1-9 are queued because any other process can queue commands (including increments) at any time (e.g., after the user space driver has written commands 2 and 4 to the buffer, but before those commands have been queued by the kernel driver).

To solve this, we add a command at the end of each packet that increments the base register the same number of times as the sync point was incremented:

10. baseA += 2

This means that, at the start of any packet of commands, the future value of syncpointA and the future value of baseA are always the same. So command 6 can be:

WAIT_SYNCPT_BASE syncpt=A base=A offset=1

and command 8 can be:

WAIT_SYNCPT_BASE syncpt=A base=A offset=2

Note: *Software never writes a new value to the syncpointA or baseA register except when the kernel driver is initialized (e.g., when the system boots). After that only the syncpoint register is incremented and only the base register is added to. All of this depends on all software that creates cooperating command buffers. Software has this policy (e.g., the user space driver always tells the kernel driver how many increments are contained in each packet of commands).*

14.6.5 Indirect/Direct Register Addressing Scheme

The indirect addressing scheme works as follows:

- An address is decoded and either routed to a channel register or to a synchronous register
- A write to INDCNT or INDDATA triggers a register access to the register specified in INDOFF
- A transaction is created and pushed into the REGF FIFO. The owning channel is specified by which INDCNT or INDDATA is written. The client and offset are specified in INDOFF.
- The transaction is routed to the target client over the HWR bus.
- If the access is a write, the transaction is complete. In the case of a read, the processor polls the channel's read return FIFO status (FIFOSTAT), waiting for a nonempty FIFO.
- The client returns the read to the channel's return FIFO.
- The processor reads INDDATA register of that particular channel to pops data from the return FIFO.

Direct addressing uses the same flow with some adjustments:

- An address is decoded and either routed to a channel register, a synchronous register, or a client
- This step is nonexistent; direct accesses do not require a trigger.
- In the case of a client decode, a transaction is created and pushed into a FIFO called REGF. The owning channel is specified as CPU_READ_RETURN_TAG. The client and offset are indicated by the address: the client is created by shifting the address down by 18; the offset is simply the lower 18 bits.
- The transaction is routed to the target client over the HWR bus. If the access is a write, the transaction is complete.
- In case of a read, client returns the read data and channel ID in the returned packet indicates its destination, be it interface or a channel's return FIFO. If the channel is 0-8, it is an indirect read and returned to the indicated channel. If the channel matches CPU_READ_RETURN_TAG(0xf), the data is returned to the IP interface.

Currently, there is support for only one pending read per interface.

14.7 Host Channel Opcodes

This section provides the format of opcodes that can be sent through the command FIFO.

HCFCMD

Generic command FIFO packet (contains fields common to all opcodes) and is used for initial decode. All command FIFO packets are multiples of 32 bits.

HCFSETCL

The SetClass opcode specifies which class is being referenced (may cause rerouting of subsequent methods/data). In addition to switching classes, the opcode allows some methods to be programmed on the switch similar to an HCFMASK opcode.

HCFINCR

The Incrementing opcode indicates the offset should be incremented for each data that is part of the packet. The count argument indicates how many 32-bit values are following. If channel protect is enabled, the host should prevent a channel switch from occurring at the end of this command packet.

HCFNONINCR

The Non-Incrementing opcode indicates the same offset should be sent for each data that is part of the packet. The count argument indicates how many 32-bit values are following. If channel protect is enabled, the host should prevent a channel switch from occurring at the end of this command packet.

HCFMASK

The Mask opcode, from the starting offset, generates offsets based on where the bits are set in the mask. The host expects the amount of data following to equal the number of bits set. If channel protect is enabled, the host should prevent a channel switch from occurring at the end of this command packet.

HCFIMM

The Immediate opcode indicates the offset and data are contained in the same 32-bit data. Only the lowest 16 bits of data are sent to the module (IMMDATA). The upper 16 bits are zeroed out.

HCFRESTART

The Restart opcode is specific to DMA operation and causes the host to set DMAGET to (ADDRESS << 4), so the next command fetch will be from (DMASTART + DMAGET).

In legacy chips, bits 27:0 were not decoded and assumed to be 0's (allowing only simply wrapping of GET back to the top of the command buffer). ADDRESS can be 0 for compatible RESTARTs or non-zero acting as a JUMP.

Note that the jump address granularity is 16 bytes, since the bottom 4 bits cannot be specified.

HCFGATHER

The Gather opcode allows contiguous portions of memory to be fetched and placed in line with the command stream, replacing the two words of the gather command. It optionally can put an incrementing or non-incrementing opcode in the stream ahead of the gathered data. This allows for the gathered data to be a pure data stream and not be required to have host opcodes inside.

HCFCHDONE

This opcode indicates to the command processor that the current channel is done processing for now and is willing to give up any of its owned modules to other channels that need them.

14.8 Host Channel Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Registers of the Host1x clients can have two offsets shown:

- "Offset" which is the Host1x offset, these generally increment by 1.
- "Byte Offset" which is the direct addressing offset for the processors, these generally increment by 4.

Other controllers in the Tegra SOC have a single "Offset", which is the same as the "Byte Offset" and used for direct addressing for the processors.

14.8.1 HOST1X_CHANNEL_FIFOSTAT_0

CFNUMEMPTY is the number of free slots available in the per-channel command. A FIFO is needed for PIO or polling for completion of a wait.

Offset: 0x0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	INDRDY: Indicates that INDCOUNT==0, so it should be OK to issue another read
28:24	X	OUTFENTRIES: Number of entries available for reading in this channel's output FIFO
20:16	X	REGFNUMEMPTY: Register write/read FIFO free count
12	X	CFGATHER: Indicates whether GATHER is active. If a GATHER command issued via PIO, software must wait for the GATHER to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
11	X	CFEMPTY: Indicates whether the command FIFO is empty or not 0 = NOTEMPTY 1 = EMPTY
10:0	X	CFNUMEMPTY: Command FIFO free count

14.8.2 HOST1X_CHANNEL_INDOFF_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
30	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
29	X	BUF32B: Buffer up 32 bits of register data before sending it. Otherwise, register writes will be sent as soon as they are received. Does not support byte writes in 16-bit host. Does not affect frame buffer writes. 0 = NOBUF 1 = BUF
28:27	X	INDSWAP: Indirect frame buffer access swap control. 00 = No byte swap 01 = 16-bit byte swap ([31:0] -> {[23:16],[31:24],[7:0],[15:8]}) 10 = 32-bit byte swap ([31:0] -> {[7:0],[15:8],[23:16],[31:24]}) 11 = 32-bit word swap ([31:0] -> {[15:8],[7:0],[31:24],[23:16]}) 0 = NONE 1 = BYTE16 2 = BYTE32 3 = WORD32
25:18	X	INDMODID: ACCTYPE=REG: Register module ID. This field should not be programmed to equal the module_id of Host1x 0 = HOST1X 1 = DPAUX1 2 = VI 4 = TSECB 8 = DISPLAY 9 = DISPLAYB 11 = TVO 12 = DSI 13 = VIC 14 = NVJPG 16 = DSIB 18 = NVDEC 19 = NVENC 20 = TSEC 21 = SOR 22 = SOR1 23 = DPAUX 24 = ISP 26 = ISPB VII2C
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
0	X	INDOFFUPD: Optionally disable the update of INDOFFSET when writing this register 0 = UPDATE 1 = NO_UPDATE

14.8.3 HOST1X_CHANNEL_INDCNT_0

Indirect Register Access Count

Used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. Only the protected channel should make liberal use of this feature for speeding up context switching.

For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	INDCOUNT

14.8.4 HOST1X_CHANNEL_INDDATA_0

This register, when written, writes to the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again. The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INDDATA: read or write data

14.8.5 HOST1X_CHANNEL_RAISE_0

The general-purpose channels have DMA and RAISE/REFCOUNT functionality.

Any raise values returned from a client module are converted to vectors and update the per-channel raise register. The RAISE vector is also writable by the CPU. Any bits set in the RAISE field when written will be set in the RAISE register, allowing any pending WAITs to continue.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	RAISE: This channel's RAISE vector

14.8.6 HOST1X_CHANNEL_DMASTART_0

This register triggers a DMA fetch from the frame buffer for this channel, if the Put register does not equal the DMA Get register.

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	DMASTART: cmdbuf frame buffer offset

14.8.7 HOST1X_CHANNEL_DMAPUT_0

This register triggers a DMA fetch from the frame buffer for this channel, if the PUT register does not equal the GET register. This address is relative to the DMASTART base address. Does not support byte writes. All 4-byte data need to be programmed.

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	DMAPUT: cmdbuf frame buffer offset

14.8.8 HOST1X_CHANNEL_DMAGET_0

This register tracks the frame-buffer offset the DMA engine has read up to (incremented as entries are loaded from the channels command buffer into the FIFO). This address is relative to the DMASTART base address.

Offset: 0x1c │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	DMAGET: cmdbuf frame buffer offset

14.8.9 HOST1X_CHANNEL_DMAEND_0

The boundary of illegal addresses (either end of push-buffer or end of physical memory). This is designed to prevent DMA from prefetching illegal addresses. If DMA reaches this address before seeing a RESTART, it will stop. This would be a software error condition.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	DMAEND: cmdbuf frame buffer offset

14.8.10 HOST1X_CHANNEL_DMACTRL_0

DMA Control Register

DMAGETRST: Reset GET pointer to '0'. Useful for cleaning up crashed channels. DMAGET is not updated instantly. Takes 4 cycles between programming of reset and a valid DMAGET.

DMAGETINIT: Reset the GET pointer to the value of DMAPUT when DMAGETRST is asserted.

DMASTOP: Stop DMA from fetching on this channel.

Note: *A Command DMA channel needs to be enabled for PIO-gather to work.*

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx001)

Bit	Reset	Description
2	0x0	DMAINITGET: Reset GET pointer to the value of DMAPUT when DMAGETRST is asserted. 0 = DISABLE 1 = ENABLE
1	0x0	DMAGETRST: Reset GET pointer to '0'. Useful for cleaning up crashed channels. 0 = DISABLE 1 = ENABLE
0	0x1	DMASTOP: Stop DMA from fetching on this channel. NOTE: a Command DMA channel needs to be enabled for PIO gather to work. 0 = RUN 1 = STOP

14.8.11 HOST1X_CHANNEL_INDOFF2_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame-buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame-buffer writes.

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID2: ACCTYPE=REG: register module ID 0 = HOST1X 1 = DPAUX1 2 = VI 4 = TSECB 8 = DISPLAY 9 = DISPLAYB11 = TVO 12 = DSI 13 = VIC 14 = NVJPG 16 = DSIB 18 = NVDEC 19 = NVENC 20 = TSEC 21 = SOR 22 = SOR1 23 = DPAUX 24 = ISP 26 = ISPB 27 = VII2C
31:2	X	INDOFFSET2: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET2: ACCTYPE=REG: register offset ([15:0])

14.8.12 HOST1X_CHANNEL_TICKCOUNT_HI_0

This register holds the high 32 bits of the tick count value.

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI

14.8.13 HOST1X_CHANNEL_TICKCOUNT_LO_0

This register holds the low 32 bits of tick count value.

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO

14.8.14 HOST1X_CHANNEL_CHANNELCTRL_0

This register will be used for controlling some channel-related commands including enabling/disabling of the tick counter, including enabling/disabling of Kernel command filtering from the gather buffers.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0)

Bit	Reset	Description
2	DISABLE	KERNEL_FILTER_GBUFFER: Enable setclass command filter for gather buffers 0 = DISABLE 1 = ENABLE
0	DISABLE	ENABLETICKCNT: enable or disable tick counter 0 = DISABLE 1 = ENABLE

14.8.15 HOST1X_CHANNEL_PAYLOAD_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PAYLOAD

14.8.16 HOST1X_CHANNEL_STALLCTRL_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_STALL

14.8.17 HOST1X_CHANNEL_STALLCOUNT_HI_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_HI

14.8.18 HOST1X_CHANNEL_STALLCOUNT_LO_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_LO

14.8.19 HOST1X_CHANNEL_XFERCTRL_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_XFER

14.8.20 HOST1X_CHANNEL_CHANNEL_XFER_HI_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_HI

14.8.21 HOST1X_CHANNEL_CHANNEL_XFER_LO_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_LO

14.8.22 HOST1X_CHANNEL_HOST1X_CHANNEL_SPARE_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:16	0xffff	CHANNEL_SPARE_HI
15:0	0x0	CHANNEL_SPARE_LO

14.9 Host SYNC Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

14.9.1 HOST1X_SYNC_INTSTATUS_0

INTSTATUS - interrupt status contains the interrupt status for all of the client modules. These status bits are only status. Writing '1' to them will not clear them. Software must clear the interrupt in the appropriate module, which should be reflected here.

Offset: 0x0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SYNCPT_CPU1_INT: set if SYNCPT_CPU1_INTSTATUS has a pending interrupt 0 = NOT_PENDING 1 = PENDING
30	X	SYNCPT_CPU0_INT: set if SYNCPT_CPU0_INTSTATUS has a pending interrupt 0 = NOT_PENDING 1 = PENDING
27	X	VII2C_INT: 0 = NOT_PENDING 1 = PENDING
26	X	ISPB_INT: 0 = NOT_PENDING 1 = PENDING
24	X	ISP_INT: 0 = NOT_PENDING 1 = PENDING
23	X	DPAUX_INT: 0 = NOT_PENDING 1 = PENDING
22	X	SOR1_INT: 0 = NOT_PENDING 1 = PENDING
21	X	SOR_INT: 0 = NOT_PENDING 1 = PENDING
20	X	TSEC_INT: 0 = NOT_PENDING 1 = PENDING
19	X	NVENC_INT: 0 = NOT_PENDING 1 = PENDING
18	X	NVDEC_INT: 0 = NOT_PENDING 1 = PENDING
16	X	DSIB_INT: 0 = NOT_PENDING 1 = PENDING
14	X	NVJPG_INT: 0 = NOT_PENDING 1 = PENDING
13	X	VIC_INT: 0 = NOT_PENDING 1 = PENDING
12	X	DSI_INT: 0 = NOT_PENDING 1 = PENDING
9	X	DISPLAYB_INT: 0 = NOT_PENDING 1 = PENDING
8	X	DISPLAY_INT: 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
4	X	TSECB_INT: 0 = NOT_PENDING 1 = PENDING
2	X	VI_INT: 0 = NOT_PENDING 1 = PENDING
1	X	DPAUX1_INT: 0 = NOT_PENDING 1 = PENDING
0	X	HOST_INT: 0 = NOT_PENDING 1 = PENDING

14.9.2 HOST1X_SYNC_INTMASK_0

Contains a master interrupt mask for all interrupt signals. If the interface's MASK_ALL bit is disabled, no interrupts will be triggered on that interface. This applies to only the non-syncpt interrupts.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx00)

Bit	Reset	Description
4	0x1	CPU_TZ_ILLEGAL_TZ_ACCESS_MASK: 0 = DISABLE 1 = ENABLE
1	0x0	CPU1_INT_MASK_ALL: 0 = DISABLE 1 = ENABLE
0	0x0	CPU0_INT_MASK_ALL: 0 = DISABLE 1 = ENABLE

14.9.3 HOST1X_SYNC_INTC0MASK_0

INTC0MASK - Interrupt Mask for CPU0

Contains the interrupt mask bits for all of the client modules. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to CPU0's interrupt signal.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00x0000000x0x000xx00xxx0x000)

Bit	Reset	Description
27	0x0	VII2C_INT_C0MASK: 0 = DISABLE 1 = ENABLE
26	0x0	ISPB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
24	0x0	ISP_INT_C0MASK: 0 = DISABLE 1 = ENABLE
23	0x0	DPAUX_INT_C0MASK: 0 = DISABLE 1 = ENABLE
22	0x0	SOR1_INT_C0MASK: 0 = DISABLE 1 = ENABLE
21	0x0	SOR_INT_C0MASK: 0 = DISABLE 1 = ENABLE
20	0x0	TSEC_INT_C0MASK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	NVENC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
18	0x0	NVDEC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
16	0x0	DSIB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
14	0x0	NVJPG_INT_C0MASK: 0 = DISABLE 1 = ENABLE
13	0x0	VIC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
12	0x0	DSI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
9	0x0	DISPLAYB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
8	0x0	DISPLAY_INT_C0MASK: 0 = DISABLE 1 = ENABLE
4	0x0	TSECB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
2	0x0	VI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
1	0x0	DPAUX1_INT_C0MASK: 0 = DISABLE 1 = ENABLE
0	0x0	HOST_INT_C0MASK: 0 = DISABLE 1 = ENABLE

14.9.4 HOST1X_SYNC_INTC1MASK_0

INTC1MASK - Interrupt Mask for CPU1

Contains the interrupt mask bits for all of the client modules. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to the CPU1's interrupt signal.

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxx00x0000000x0x000xx00xxx0x000)

Bit	Reset	Description
27	0x0	VII2C_INT_C1MASK: 0 = DISABLE 1 = ENABLE
26	0x0	ISPB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
24	0x0	ISP_INT_C1MASK: 0 = DISABLE 1 = ENABLE
23	0x0	DPAUX_INT_C1MASK: 0 = DISABLE 1 = ENABLE
22	0x0	SOR1_INT_C1MASK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21	0x0	SOR_INT_C1MASK: 0 = DISABLE 1 = ENABLE
20	0x0	TSEC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
19	0x0	NVENC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
18	0x0	NVDEC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
16	0x0	DSIB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
14	0x0	NVJPG_INT_C1MASK: 0 = DISABLE 1 = ENABLE
13	0x0	VIC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
12	0x0	DSI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
9	0x0	DISPLAYB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
8	0x0	DISPLAY_INT_C1MASK: 0 = DISABLE 1 = ENABLE
4	0x0	TSECB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
2	0x0	VI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
1	0x0	DPAUX1_INT_C1MASK: 0 = DISABLE 1 = ENABLE
0	0x0	HOST_INT_C1MASK: 0 = DISABLE 1 = ENABLE

14.9.5 HOST1X_SYNC_HINTSTATUS_0

Host Interrupt Status

Contains the interrupt status for the various host interrupts. The status is sticky and is PENDING until cleared (write 1's to INTSTATUS to clear).

Offset: 0x20 │ Read/Write: R/W │ Reset: 0xX0000000 (0bx000xxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31	RO	X	HINTSTATUS_EXT_INT: Additional interrupts pending in the HINSTATUS_EXT register 0 = NOT_PENDING 1 = PENDING
30	RW	0x0	TIMER_INTP: Timer Interrupt from the Protected Channel 0 = NOT_PENDING 1 = PENDING
29	RW	0x0	CSW_HOST1XW2MC_INT: Host write client FIFO has filled up

Bit	R/W	Reset	Description
28	RW	0x0	XBAR_TSEC_TIMEOUT_ID: Id of the CPU which timed out and updated the timeout address 0 = TSEC 1 = XBAR
21	RW	0x0	WAIT_INT13: WAIT has completed on channel 13 0 = NOT_PENDING 1 = PENDING
20	RW	0x0	WAIT_INT12: WAIT has completed on channel 12 0 = NOT_PENDING 1 = PENDING
19	RW	0x0	RDMA_DATABUF_THOLD_INT0: Read DMA data FIFO in port0 reached high level watermark 0 = NOT_PENDING 1 = PENDING
18	RW	0x0	RDMA_BUF_THOLD_INT0: Buffer threshold reached in read DMA port0 0 = NOT_PENDING 1 = PENDING
17	RW	0x0	RDMA_BUF_OFLOW_INT0: Buffer overflow in read DMA port0 0 = NOT_PENDING 1 = PENDING
16	RW	0x0	RDMA_INVAL_CLREQ_INT: Invalid client request to read DMA 0 = NOT_PENDING 1 = PENDING
15	RW	0x0	UNIT4_ACTMON_INTR: Unit4 Actmon generates interrupt 0 = NOT_PENDING 1 = PENDING
14	RW	0x0	UNIT3_ACTMON_INTR: Unit3 Actmon generates interrupt 0 = NOT_PENDING 1 = PENDING
13	RW	0x0	UNIT2_ACTMON_INTR: Unit2 ACTMON generates interrupt 0 = NOT_PENDING 1 = PENDING
12	RW	0x0	UNIT1_ACTMON_INTR: Unit1 ACTMON generates interrupt 0 = NOT_PENDING 1 = PENDING
11	RW	0x0	WAIT_INT11: WAIT has completed on channel 11 0 = NOT_PENDING 1 = PENDING
10	RW	0x0	WAIT_INT10: WAIT has completed on channel 10 0 = NOT_PENDING 1 = PENDING
9	RW	0x0	WAIT_INT9: WAIT has completed on channel 9 0 = NOT_PENDING 1 = PENDING
8	RW	0x0	WAIT_INT8: WAIT has completed on channel 8 0 = NOT_PENDING 1 = PENDING
7	RW	0x0	WAIT_INT7: WAIT has completed on channel 7 0 = NOT_PENDING 1 = PENDING
6	RW	0x0	WAIT_INT6: WAIT has completed on channel 6 0 = NOT_PENDING 1 = PENDING
5	RW	0x0	WAIT_INT5: WAIT has completed on channel 5 0 = NOT_PENDING 1 = PENDING
4	RW	0x0	WAIT_INT4: WAIT has completed on channel 4 0 = NOT_PENDING 1 = PENDING
3	RW	0x0	WAIT_INT3: WAIT has completed on channel 3 0 = NOT_PENDING 1 = PENDING

Bit	R/W	Reset	Description
2	RW	0x0	WAIT_INT2: WAIT has completed on channel 2 0 = NOT_PENDING 1 = PENDING
1	RW	0x0	WAIT_INT1: WAIT has completed on channel 1 0 = NOT_PENDING 1 = PENDING
0	RW	0x0	WAIT_INT0: WAIT has completed on channel 0 0 = NOT_PENDING 1 = PENDING

14.9.6 HOST1X_SYNC_HINTMASK_0

Host Interrupt Mask

Contains the interrupt mask bits for all of the host interrupts. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to the global interrupt signal.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	HINTSTATUS_EXT_INTMASK: 0 = DISABLE 1 = ENABLE
30	0x0	TIMER_INTMASKP: Timer Interrupt Mask for the Protected Channel 0 = DISABLE 1 = ENABLE
29	0x0	CSW_HOST1XW2MC_INTMASK: 0 = DISABLE 1 = ENABLE
21	0x0	WAIT_INTMASK13: 0 = DISABLE 1 = ENABLE
20	0x0	WAIT_INTMASK12: 0 = DISABLE 1 = ENABLE
19	0x0	RDMA_DATABUF_THOLD_INTMASK0: 0 = DISABLE 1 = ENABLE
18	0x0	RDMA_BUF_THOLD_INTMASK0: 0 = DISABLE 1 = ENABLE
17	0x0	RDMA_BUF_OFLOW_INTMASK0: 0 = DISABLE 1 = ENABLE
16	0x0	RDMA_INVAL_CLREQ_INTMASK: 0 = DISABLE 1 = ENABLE
15	0x0	UNIT4_ACTMON_INTRMASK: 0 = DISABLE 1 = ENABLE
14	0x0	UNIT3_ACTMON_INTRMASK: 0 = DISABLE 1 = ENABLE
13	0x0	UNIT2_ACTMON_INTRMASK: 0 = DISABLE 1 = ENABLE
12	0x0	UNIT1_ACTMON_INTRMASK: 0 = DISABLE 1 = ENABLE
11	0x0	WAIT_INTMASK11: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x0	WAIT_INTMASK10: 0 = DISABLE 1 = ENABLE
9	0x0	WAIT_INTMASK9: 0 = DISABLE 1 = ENABLE
8	0x0	WAIT_INTMASK8: 0 = DISABLE 1 = ENABLE
7	0x0	WAIT_INTMASK7: 0 = DISABLE 1 = ENABLE
6	0x0	WAIT_INTMASK6: 0 = DISABLE 1 = ENABLE
5	0x0	WAIT_INTMASK5: 0 = DISABLE 1 = ENABLE
4	0x0	WAIT_INTMASK4: 0 = DISABLE 1 = ENABLE
3	0x0	WAIT_INTMASK3: 0 = DISABLE 1 = ENABLE
2	0x0	WAIT_INTMASK2: 0 = DISABLE 1 = ENABLE
1	0x0	WAIT_INTMASK1: 0 = DISABLE 1 = ENABLE
0	0x0	WAIT_INTMASK0: 0 = DISABLE 1 = ENABLE

14.9.7 HOST1X_SYNC_HINTSTATUS_EXT_0

Extended Host Interrupt Status

Contains additional interrupt status bits that did not fit in the HINTSTATUS register. When any of these bits is set, the HINTSTATUS_EXT_INT bit will also be set in the HINSTATUS register. Each status bit is sticky and PENDING until cleared (write 1's to HINTSTATUS_EXT to clear).

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
31	0x0	IP_WRITE_INT: Write transaction timeout occurred after IP_BUSY_TIMEOUT cycles. Offending address in IP_WRITE_TIMEOUT_ADDR. 0 = NOT_PENDING 1 = PENDING
30	0x0	IP_READ_INT: Read transaction timeout occurred after IP_BUSY_TIMEOUT cycles. Offending address in IP_READ_TIMEOUT_ADDR. 0 = NOT_PENDING 1 = PENDING
13	0x0	CMDPP_ILLEGAL_OPCODE_INT13: CMDPP13 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
12	0x0	CMDPP_ILLEGAL_OPCODE_INT12: CMDPP12 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
11	0x0	CMDPP_ILLEGAL_OPCODE_INT11: CMDPP11 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
10	0x0	CMDPP_ILLEGAL_OPCODE_INT10: CMDPP10 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
9	0x0	CMDPP_ILLEGAL_OPCODE_INT9: CMDPP9 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
8	0x0	CMDPP_ILLEGAL_OPCODE_INT8: CMDPP8 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
7	0x0	CMDPP_ILLEGAL_OPCODE_INT7: CMDPP7 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
6	0x0	CMDPP_ILLEGAL_OPCODE_INT6: CMDPP6 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
5	0x0	CMDPP_ILLEGAL_OPCODE_INT5: CMDPP5 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
4	0x0	CMDPP_ILLEGAL_OPCODE_INT4: CMDPP4 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
3	0x0	CMDPP_ILLEGAL_OPCODE_INT3: CMDPP3 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
2	0x0	CMDPP_ILLEGAL_OPCODE_INT2: CMDPP2 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
1	0x0	CMDPP_ILLEGAL_OPCODE_INT1: CMDPP1 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
0	0x0	CMDPP_ILLEGAL_OPCODE_INT0: CMDPP0 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING

14.9.8 HOST1X_SYNC_HINTMASK_EXT_0

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
31	0x0	IP_WRITE_INTMASK: 0 = DISABLE 1 = ENABLE
30	0x0	IP_READ_INTMASK: 0 = DISABLE 1 = ENABLE
13	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK13: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
12	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK12: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
11	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK11: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
10	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK10: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK9: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
8	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK8: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
7	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK7: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
6	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK6: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
5	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK5: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
4	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK4: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
3	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK3: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
2	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK2: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
1	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK1: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
0	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK0: Mask CMDPP_ILLEGAL_OPCODE_INT0 interrupt bit. 0 = DISABLE 1 = ENABLE

14.9.9 HOST1X_SYNC_CF_SETUPDONE_0

Write to this register to trigger an update of the FIFO pointers. **ONLY DO THIS WHEN THE FIFO IS EMPTY.**

Offset: 0xa0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CF_SETUPDONE: Dummy bit

14.9.10 HOST1X_SYNC_CMDPROC_CTRL_0

Offset: 0xa4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx00x)

Bit	Reset	Description
5	0x0	INTFC_CLKEN_OVR
2	0x0	GATHER_PARSE_DISABLED
1	0x0	DROP_ILLEGAL_OPCODES

14.9.11 HOST1X_SYNC_CMDPROC_STAT_0

Offset: 0xa8 │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:0	X	ILLEGAL_OPCODE

14.9.12 HOST1X_SYNC_CMDPROC_STOP_0

CH*_CMDPROC_STOP stops issuing commands from the command FIFO. This is useful to stop other channels when a channel teardown is needed to prevent unwanted traffic from happening at the same time.

Offset: 0xac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13	0x0	CH13_CMDPROC_STOP: 0 = RUN 1 = STOP
12	0x0	CH12_CMDPROC_STOP: 0 = RUN 1 = STOP
11	0x0	CH11_CMDPROC_STOP: 0 = RUN 1 = STOP
10	0x0	CH10_CMDPROC_STOP: 0 = RUN 1 = STOP
9	0x0	CH9_CMDPROC_STOP: 0 = RUN 1 = STOP
8	0x0	CH8_CMDPROC_STOP: 0 = RUN 1 = STOP
7	0x0	CH7_CMDPROC_STOP: 0 = RUN 1 = STOP
6	0x0	CH6_CMDPROC_STOP: 0 = RUN 1 = STOP
5	0x0	CH5_CMDPROC_STOP: 0 = RUN 1 = STOP
4	0x0	CH4_CMDPROC_STOP: 0 = RUN 1 = STOP
3	0x0	CH3_CMDPROC_STOP: 0 = RUN 1 = STOP
2	0x0	CH2_CMDPROC_STOP: 0 = RUN 1 = STOP
1	0x0	CH1_CMDPROC_STOP: 0 = RUN 1 = STOP
0	0x0	CH0_CMDPROC_STOP: 0 = RUN 1 = STOP

14.9.13 HOST1X_SYNC_CH_TEARDOWN_0

Channel teardown register. Tells the hardware that a channel has gone away. Will reset that channel's command FIFO and release any locks it has in the arbiter. Will NOT reset that channel's output FIFO, which can be emptied by reading out all remaining entries.

This is a write-only register. A read of this register will yield all 0s.

Offset: 0xb0 | Read/Write: W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13	X	CH13_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
12	X	CH12_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
11	X	CH11_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
10	X	CH10_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
9	X	CH9_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
8	X	CH8_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
7	X	CH7_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
6	X	CH6_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
5	X	CH5_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
4	X	CH4_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
3	X	CH3_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
2	X	CH2_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
1	X	CH1_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
0	X	CH0_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

14.9.14 HOST1X_SYNC_MOD_TEARDOWN_0

Module teardown register. If a module is reset, the host needs to reset its state with respect to which channels own that module. Whenever a module is reset, the corresponding teardown bit in this register should be written. Module teardown will only work if the command FIFO and command processor are in a good state (the FIFO is empty of traffic for that channel and the command processor is at an opcode boundary). If an entire channel needs to be reset, the CH_TEARDOWN register should be used instead.

Offset: 0xb4 | Read/Write: R/W | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
27	X	VII2C_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
26	X	ISPB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

Bit	Reset	Description
24	X	ISP_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
23	X	DPAUX_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
22	X	SOR1_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
21	X	SOR_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
20	X	TSEC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
19	X	NVENC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
18	X	NVDEC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
16	X	DSIB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
14	X	NVJPG_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
13	X	VIC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
12	X	DSI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
9	X	DISPLAYB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
8	X	DISPLAY_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
4	X	TSECB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
2	X	VI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
1	X	DPAUX1_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

14.9.15 HOST1X_SYNC_DISPLAY_STATUS_0

The per-client status registers indicate which channel owns each client as well as the current working class for each client. This is useful for determining each client's state with regard to granting context switches. _CURRCL is the current class ID for that module.

Offset: 0xdc │ Read/Write: RO │ Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DISPLAY_CURRCL

14.9.16 HOST1X_SYNC_DISPLAYB_STATUS_0

Offset: 0xe0 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DISPLAYB_CURRCL

14.9.17 HOST1X_SYNC_ISP_STATUS_0

Offset: 0xe4 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	ISP_CURRCL

14.9.18 HOST1X_SYNC_DSI_STATUS_0

Offset: 0xe8 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DSI_CURRCL

14.9.19 HOST1X_SYNC_SOR1_STATUS_0

Offset: 0xec | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	SOR1_CURRCL

14.9.20 HOST1X_SYNC_SOR_STATUS_0

Offset: 0xf0 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	SOR_CURRCL

14.9.21 HOST1X_SYNC_DPAUX_STATUS_0

Offset: 0xf4 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DPAUX_CURRCL

14.9.22 HOST1X_SYNC_VI_STATUS_0

Offset: 0xf8 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	VI_CURRCL

14.9.23 HOST1X_SYNC_DSIB_STATUS_0

Offset: 0xfc | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DSIB_CURRCL

14.9.24 HOST1X_SYNC_VIC_STATUS_0

Offset: 0x100 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	VIC_CURRCL

14.9.25 HOST1X_SYNC_NVENC_STATUS_0

Offset: 0x104 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	NVENC_CURRCL

14.9.26 HOST1X_SYNC_TSEC_STATUS_0

Offset: 0x108 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	TSEC_CURRCL

14.9.27 HOST1X_SYNC_ISPB_STATUS_0

Offset: 0x10c | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	ISPB_CURRCL

14.9.28 HOST1X_SYNC_NVDEC_STATUS_0

Offset: 0x110 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	NVDEC_CURRCL

14.9.29 HOST1X_SYNC_NVJPG_STATUS_0

Offset: 0x114 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	NVJPG_CURRCL

14.9.30 HOST1X_SYNC_VII2C_STATUS_0

Offset: 0x118 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	VII2C_CURRCL

14.9.31 HOST1X_SYNC_DPAUX1_STATUS_0

Offset: 0x11c | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DPAUX1_CURRCL

14.9.32 HOST1X_SYNC_TSECB_STATUS_0

Offset: 0x120 | Read/Write: RO | Reset: 0x0XXX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	TSECB_CURRCL

14.9.33 HOST1X_SYNC_DIRECT_MODULE_CONFIG_0

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x54000000 (0b01010100000000000000000000000xx)

Bit	Reset	Description
31:2	0x15000000	BASE

14.9.34 HOST1X_SYNC_USEC_CLK_0

Number of host clocks needed to make a microsecond. Used for the DELAY host method. For example, if the host clock is 250 MHz, this register should be programmed to 250. If the host clock is 150 MHz, this register should be programmed to 150.

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x0000015e (0bxxxxxxxxxxxxxxxxxxxx000101011110)

Bit	Reset	Description
11:0	0x15e	USEC_CLKS

14.9.35 HOST1X_SYNC_CTXSW_TIMEOUT_CFG_0

When a channel writes to a new class on a client, it generates a context switch. To keep from continually switching contexts if channels are addressing the same client, this register is used, so that the Host1x does not switch until the channel has either received WAIT_CTXSW_CNT clocks before switching or the channel stops targeting the particular client. If two channels are accessing two classes of a client, to guarantee a channel sending multiple commands for a class, this register can be used.

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxx00001111)

Bit	Reset	Description
7:0	0xf	WAIT_CTXSW_CNT: Number of cycles to wait

14.9.36 HOST1X_SYNC_INDREG_DMA_CTRL_0

Enable use of DMA engine to move data from indirect register interface to memory. ATTN_LVL controls flow between host and DMA. DMA will not start a transfer until the set number of slots are full:

- 00 = 1 slot
- 01 = 4 slots
- 10 = 8 slots

Offset: 0x1ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000x0000)

Bit	Reset	Description
7	0x0	AHBDMA_ENABLE: Enable generation of request to DMA engine 0 = DISABLE 1 = ENABLE
6:5	0x0	AHBDMA_ATTN_LVL: Number of entries to receive before sending DMA request, 1, 4, or 8
3:0	0x0	AHBDMA_CHID: channel being used by indirect read for DMA (which chout FIFO to monitor)

14.9.37 HOST1X_SYNC_CHANNEL_PRIORITY_0

Set channel priority for dual ring arbitration (hi/lo). Used in arbitrating MLOCKs.

Offset: 0x1b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13	0x0	HIPRI_CH13: 0 = DISABLE 1 = ENABLE
12	0x0	HIPRI_CH12: 0 = DISABLE 1 = ENABLE
11	0x0	HIPRI_CH11: 0 = DISABLE 1 = ENABLE
10	0x0	HIPRI_CH10: 0 = DISABLE 1 = ENABLE
9	0x0	HIPRI_CH9: 0 = DISABLE 1 = ENABLE
8	0x0	HIPRI_CH8: 0 = DISABLE 1 = ENABLE
7	0x0	HIPRI_CH7: 0 = DISABLE 1 = ENABLE
6	0x0	HIPRI_CH6: 0 = DISABLE 1 = ENABLE
5	0x0	HIPRI_CH5: 0 = DISABLE 1 = ENABLE
4	0x0	HIPRI_CH4: 0 = DISABLE 1 = ENABLE
3	0x0	HIPRI_CH3: 0 = DISABLE 1 = ENABLE
2	0x0	HIPRI_CH2: 0 = DISABLE 1 = ENABLE
1	0x0	HIPRI_CH1: 0 = DISABLE 1 = ENABLE
0	0x0	HIPRI_CH0: 0 = DISABLE 1 = ENABLE

14.9.38 HOST1X_SYNC_CDMA_ASM_TIMEOUT_0

Timeout value determines how long the assembly logic will wait before it flushes data from the data FIFO to avoid head-of-line blocking.

Note: *Do not use a value of zero -- causes immediate flushing so no forward progress is made.*

Offset: 0x1b4 | Read/Write: R/W | Reset: 0x00000404 (0bxxxxxxxxxxxxxxxxxxxx00100xxx00100)

Bit	Reset	Description
12:8	0x4	CDMA_ASM_GFIFO_TIMEOUT
4:0	0x4	CDMA_ASM_DFIFO_TIMEOUT

14.9.39 HOST1X_SYNC_CDMA_MISC_0

CDMA_DELAY_PUT

Delay put_addr updates. This can potentially increase MC performance slightly by gathering multiple consecutive put_addr updates into 1 update, which reduces the amount of requests to the MC. In addition, it can be used to cover a race condition where a write to memory has not completed before CDMA starts fetching the data. But in this case, the counter has to be set to a high value, which *will* affect performance.

CDMA_SIMPLE_PREFETCH

Reduce per-channel requests to 1 every 3 cycles. This can reduce performance in 2 ways:

- The memory request FIFO will fill up slightly slower, but the impact will be minimal.
- It will cause the channel arbiter to switch to other channels when they have a request available, which will reduce locality. On the plus side, it has much less nasty corner cases.

CDMA_PUT_SYNC_DISABLE

CDMA_EN_STATS

Enable statistics counters.

- - Count the number of 128-bit words fetched by CDMA
- - Count the number of 128-bit words thrown away by CDMA

This allows us to get an idea how efficient the CDMA really is when there are multiple push-buffers active at the same time.

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0x000xx000000)

Bit	Reset	Description
12	0x0	CDMA_EN_STATS
10	0x0	CDMA_CLKEN_OVR
9	0x0	CDMA_PUT_SYNC_DISABLE
8	0x0	CDMA_SIMPLE_PREFETCH
5:0	0x0	CDMA_DELAY_PUT

14.9.40 HOST1X_SYNC_IP_BUSY_TIMEOUT_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	IP_BUSY_TIMEOUT: Number of busy cycles before requesting a retry. 0 = disabled

14.9.41 HOST1X_SYNC_IP_READ_TIMEOUT_ADDR_0

Offset: 0x1c0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	IP_READ_TIMEOUT_ADDR: Address of transaction that caused an AXI read timeout

14.9.42 HOST1X_SYNC_IP_WRITE_TIMEOUT_ADDR_0

Offset: 0x1c4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	IP_WRITE_TIMEOUT_ADDR: Address of transaction that caused an AXI write timeout

14.9.43 HOST1X_SYNC_MCCIF_THCTRL_0

Memory write client FIFO status. Reads out the available 128-bit entries. If writing through the buffered frame buffer write region, writes are accumulated up to 128 bits before being flushed, so 10 entries can mean up to 40 writes. Memory client high-priority threshold control register. Sets the threshold of when the client becomes high priority.

Offset: 0x1d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000000xxxxxxxx)

Bit	R/W	Reset	Description
13:8	RW	0x0	CSW_HOST1XW2MC_HPTH
5:0	RO	X	CSW_HOST1XW_FIFOSTAT

14.9.44 HOST1X_SYNC_HC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Note: *This FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.*

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field results in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000xxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	HC_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	HC_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	HC_CCLK_OVERRIDE
17	0x0	HC_RCLK_OVERRIDE
16	0x0	HC_WCLK_OVERRIDE
3	DISABLE	HC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	HC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	HC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	HC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

14.9.45 HOST1X_SYNC_TIMEOUT_WCOAL_HC_0

Write Coalescing Time-Out Register

Note: *Write coalescing is no longer supported by the MCCIF clients. Registers are retained for software compatibility but are not used by the hardware.*

Offset: 0x1e0 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	HOST1XW_WCOAL_TMVAL

14.9.46 HOST1X_SYNC_MLOCK_0_0

MLOCK Registers

MLOCK is 1 whenever someone holds the lock, and is 0 otherwise. MLOCK is normally set and cleared using the host methods ACQUIRE_MLOCK and RELEASE_MLOCK.

If a CPU wants to acquire a lock, then it reads this register. If the value returned is 0, it has successfully acquired the MLOCK. A return value of 1 indicates that the acquire failed. At any time, the CPU can write 0 to MLOCK, releasing the MLOCK (note that hardware does not check if the CPU owned the lock). If the CPU writes a 1 to MLOCK, this is undefined and is ignored by the hardware (a NOP). Use MLOCK_OWNER to read the status of a lock (since reading MLOCK itself has a side effect).

Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_0

14.9.47 HOST1X_SYNC_MLOCK_1_0

Offset: 0x2c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_1

14.9.48 HOST1X_SYNC_MLOCK_2_0

Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_2

14.9.49 HOST1X_SYNC_MLOCK_3_0

Offset: 0x2cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_3

14.9.50 HOST1X_SYNC_MLOCK_4_0

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_4

14.9.51 HOST1X_SYNC_MLOCK_5_0

Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_5

14.9.52 HOST1X_SYNC_MLOCK_6_0

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_6

14.9.53 HOST1X_SYNC_MLOCK_7_0

Offset: 0x2dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_7

14.9.54 HOST1X_SYNC_MLOCK_8_0

Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_8

14.9.55 HOST1X_SYNC_MLOCK_9_0

Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_9

14.9.56 HOST1X_SYNC_MLOCK_10_0

Offset: 0x2e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_10

14.9.57 HOST1X_SYNC_MLOCK_11_0

Offset: 0x2ec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_11

14.9.58 HOST1X_SYNC_MLOCK_12_0

Offset: 0x2f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_12

14.9.59 HOST1X_SYNC_MLOCK_13_0

Offset: 0x2f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_13

14.9.60 HOST1X_SYNC_MLOCK_14_0

Offset: 0x2f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_14

14.9.61 HOST1X_SYNC_MLOCK_15_0

Offset: 0x2fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_15

14.9.62 HOST1X_SYNC_MLOCK_OWNER_0_0

MLOCK_OWNER is a read-only status for MLOCK. When MLOCK_*_OWNS are all zeros, it indicates that the MLOCK is free (zero). When MLOCK has been acquired (set), then one of MLOCK_*_OWNS will be non-zero.

Either bit 0 or 1 will be set -- indicating whether a channel or a CPU has acquired the MLOCK. If a channel owns the MLOCK, then the channel number is given by the MLOCK_OWNER_CHID field.

Offset: 0x340 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_0
1	X	MLOCK_CPU_OWNS_0
0	X	MLOCK_CH_OWNS_0

14.9.63 HOST1X_SYNC_MLOCK_OWNER_1_0

Offset: 0x344 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_1
1	X	MLOCK_CPU_OWNS_1
0	X	MLOCK_CH_OWNS_1

14.9.64 HOST1X_SYNC_MLOCK_OWNER_2_0

Offset: 0x348 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_2
1	X	MLOCK_CPU_OWNS_2
0	X	MLOCK_CH_OWNS_2

14.9.65 HOST1X_SYNC_MLOCK_OWNER_3_0

Offset: 0x34c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_3
1	X	MLOCK_CPU_OWNS_3
0	X	MLOCK_CH_OWNS_3

14.9.66 HOST1X_SYNC_MLOCK_OWNER_4_0

Offset: 0x350 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_4
1	X	MLOCK_CPU_OWNS_4
0	X	MLOCK_CH_OWNS_4

14.9.67 HOST1X_SYNC_MLOCK_OWNER_5_0

Offset: 0x354 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_5
1	X	MLOCK_CPU_OWNS_5
0	X	MLOCK_CH_OWNS_5

14.9.68 HOST1X_SYNC_MLOCK_OWNER_6_0

Offset: 0x358 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_6
1	X	MLOCK_CPU_OWNS_6
0	X	MLOCK_CH_OWNS_6

14.9.69 HOST1X_SYNC_MLOCK_OWNER_7_0

Offset: 0x35c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_7
1	X	MLOCK_CPU_OWNS_7
0	X	MLOCK_CH_OWNS_7

14.9.70 HOST1X_SYNC_MLOCK_OWNER_8_0

Offset: 0x360 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_8
1	X	MLOCK_CPU_OWNS_8
0	X	MLOCK_CH_OWNS_8

14.9.71 HOST1X_SYNC_MLOCK_OWNER_9_0

Offset: 0x364 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_9
1	X	MLOCK_CPU_OWNS_9
0	X	MLOCK_CH_OWNS_9

14.9.72 HOST1X_SYNC_MLOCK_OWNER_10_0

Offset: 0x368 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_10
1	X	MLOCK_CPU_OWNS_10
0	X	MLOCK_CH_OWNS_10

14.9.73 HOST1X_SYNC_MLOCK_OWNER_11_0

Offset: 0x36c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_11
1	X	MLOCK_CPU_OWNS_11
0	X	MLOCK_CH_OWNS_11

14.9.74 HOST1X_SYNC_MLOCK_OWNER_12_0

Offset: 0x370 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_12
1	X	MLOCK_CPU_OWNS_12
0	X	MLOCK_CH_OWNS_12

14.9.75 HOST1X_SYNC_MLOCK_OWNER_13_0

Offset: 0x374 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_13
1	X	MLOCK_CPU_OWNS_13
0	X	MLOCK_CH_OWNS_13

14.9.76 HOST1X_SYNC_MLOCK_OWNER_14_0

Offset: 0x378 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_14
1	X	MLOCK_CPU_OWNS_14
0	X	MLOCK_CH_OWNS_14

14.9.77 HOST1X_SYNC_MLOCK_OWNER_15_0

Offset: 0x37c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_15
1	X	MLOCK_CPU_OWNS_15
0	X	MLOCK_CH_OWNS_15

14.9.78 HOST1X_SYNC_MLOCK_ERROR_0_0

If a channel attempts to release an MLOCK that it does not own, then the release has no effect on MLOCK, but the corresponding error bit is set (this includes releasing an MLOCK owned by no one).

Offset: 0x3c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	0x0	MLOCK_ERROR_15
14	0x0	MLOCK_ERROR_14
13	0x0	MLOCK_ERROR_13
12	0x0	MLOCK_ERROR_12
11	0x0	MLOCK_ERROR_11
10	0x0	MLOCK_ERROR_10
9	0x0	MLOCK_ERROR_9
8	0x0	MLOCK_ERROR_8
7	0x0	MLOCK_ERROR_7
6	0x0	MLOCK_ERROR_6
5	0x0	MLOCK_ERROR_5
4	0x0	MLOCK_ERROR_4
3	0x0	MLOCK_ERROR_3
2	0x0	MLOCK_ERROR_2
1	0x0	MLOCK_ERROR_1
0	0x0	MLOCK_ERROR_0

14.9.79 HOST1X_SYNC_SYNCPT_BASE_n_0

Syncpt base registers are used by wait_syncpt_base method. The wait will be released when SYNCPT[indx] >= (BASE[base_indx] + offset), where indx, base_indx, and offset are supplied by the wait method.

There are 64 Syncpt base registers, where n = 0 through 63.

Offset: 0x600 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_n

14.10 Host Class Methods

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Class offsets are always relative to the class (based at 0).

14.10.1 NV_CLASS_HOST_INCR_SYNCPT_0

All Classes have the INCR_SYNCPT method. For host, this method, immediately increments SYNCPT[indx], irrespective of the condition. Note that INCR_SYNCPT_CNTRL and INCR_SYNCPT_ERROR are included for consistency with host clients, but writes to INCR_SYNCPT_CNTRL have no effect on the operation of Host1x, and because there are no condition FIFOs to overflow, INCR_SYNCPT_ERROR will never be set.

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	INDX: syncpt index value

14.10.2 NV_CLASS_HOST_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

14.10.3 NV_CLASS_HOST_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.10.4 NV_CLASS_HOST_WAIT_SYNCPT_0

Wait on syncpt method.

Command dispatch will stall until:

SYNCPT[indx][NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0] >=
threshold[NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0]

The comparison takes into account the possibility of wrapping. Note that more bits are allocated for index and threshold than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts, and NV_HOST1X_SYNCPT_THRESH_WIDTH for the number of bits used by the comparison.

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX
23:0	X	THRESH

14.10.5 NV_CLASS_HOST_WAIT_SYNCPT_BASE_0

Wait on syncpt method using base register.

Command dispatch will stall until:

SYNCPT[indx][NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0] >= (SYNCPT_BASE[base_indx]+offset)

The comparison takes into account the possibility of wrapping. Note that more bits are allocated for INDX and BASE_INDX than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts, Use NV_HOST1X_SYNCPT_NB_BASES for the number of syncpt_bases, and NV_HOST1X_SYNCPT_THRESH_WIDTH for the number of bits used by the comparison If NV_HOST1X_SYNCPT_THRESH_WIDTH is greater than 16, the offset is sign-extended before it is added to SYNCPT_BASE.

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX
23:16	X	BASE_INDX
15:0	X	OFFSET

14.10.6 NV_CLASS_HOST_WAIT_SYNCPT_INCR_0

Wait on syncpt increment method.

Command dispatch will stall until the next time that SYNCPT[indx] is incremented.

Note that more bits are allocated for INDX than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts.

Offset: 0xa | Byte Offset: 0x28 | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX

14.10.7 NV_CLASS_HOST_LOAD_SYNCPT_BASE_0

Load syncpt base method.

SYNCPT_BASE[indx] = value

Offset: 0xb | Byte Offset: 0x2c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BASE_INDX
23:0	X	VALUE

14.10.8 NV_CLASS_HOST_INCR_SYNCPT_BASE_0

Increment syncpt base method.

SYNCPT_BASE[indx] += offset

Offset: 0xc | Byte Offset: 0x30 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BASE_INDX
23:0	X	OFFSET

14.10.9 NV_CLASS_HOST_CLEAR_0

Clear method. Any bits set in VECTOR will be cleared in the channel's RAISE vector.

Offset: 0xd | Byte Offset: 0x34 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

14.10.10 NV_CLASS_HOST_WAIT_0

Wait method. Command dispatch will stall until any of the bits set in VECTOR become set in the channel's RAISE vector.

Offset: 0xe | Byte Offset: 0x38 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

14.10.11 NV_CLASS_HOST_WAIT_WITH_INTR_0

Wait with Interrupt method. Identical to the WAIT method except an interrupt will be triggered when the WAIT requirement is satisfied. This is obsolete and preserved here only for completeness.

Offset: 0xf | Byte Offset: 0x3c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

14.10.12 NV_CLASS_HOST_DELAY_USEC_0

Delay number of microseconds. Command dispatch will stall until the number of microseconds indicated in NUSEC has passed. The timing of microseconds is controlled by the USEC_CLK register.

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:0	X	NUSEC: Enough for 1.05 seconds

14.10.13 NV_CLASS_HOST_TICKCOUNT_HI_0

This register value will initialize the high 32 bits of the tick count value in the host clock counter.

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI: Read or write tick count

14.10.14 NV_CLASS_HOST_TICKCOUNT_LO_0

This register value will initialize the low 32 bits of the tick count value in the host clock counter.

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO: Read or write tick count

14.10.15 NV_CLASS_HOST_TICKCTRL_0

This register write enables the tick counter on the host clock to start counting.

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TICKCNT_ENABLE: Enable or Disable tick counter 0 = DISABLE 1 = ENABLE

14.10.16 NV_CLASS_HOST_INDCTRL_0

Indirect Addressing

These registers (along with INDDATA) are used to indirectly read/write either register or memory. Host registers are not accessible using this interface. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA.

Either INDCTRL/INDOFF2 or INDOFF can be used, but INDOFF may not be able to address all memory in chips with large memory maps. The redundant bits in INDCTRL and INDOFF are shared, so writing either offset sets those bits.

Note: *The following restrictions apply to the use of indirect memory writes: 1) At initialization time, do a dummy indirect write (with all byte enables set to zero). 2) Dedicate an MLOCK for indirect memory writes, then before a channel issues a set of indirect memory writes it must acquire this MLOCK; after the writes have been issued, the MLOCK is released -- this will restrict the use of indirect memory writes to a single channel at a time.*

Offset: 0x2b | Byte Offset: 0xac | Read/Write: R/W | Reset: 0xXX00000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	INDBE: Byte enables. Will apply to all subsequent data transactions. Not applicable for reads.
27	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
26	X	SPOOL: Route return data to spool FIFO, only applicable to reads 0 = DISABLE 1 = ENABLE
1	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
0	X	RWN: Read/write 0 = WRITE 1 = READ

14.10.17 NV_CLASS_HOST_INDOFF2_0

Offset: 0x2c | Byte Offset: 0xb0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID: ACCTYPE=REG: Register module ID 0 = HOST1X 1 = DPAUX1 2 = VI 4 = TSECB 8 = DISPLAY 9 = DISPLAYB 11 = TVO 12 = DSI 13 = VIC 14 = NVJPG 16 = DSIB 18 = NVDEC 19 = NVENC 20 = TSEC 21 = SOR 22 = SOR1 23 = DPAUX 24 = ISP 26 = ISPB 27 = VII2C
31:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])

14.10.18 NV_CLASS_HOST_INDOFF_0

Offset: 0x2d | Byte Offset: 0xb4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	INDBE: Byte enables. Will apply to all subsequent data transactions. Not applicable for reads.
27	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
26	X	SPOOL: Route return data to spool FIFO, only applicable to reads 0 = DISABLE 1 = ENABLE
25:18	X	INDMODID: ACCTYPE=REG: register module ID 0 = HOST1X 1 = DPAUX1 2 = VI 4 = TSECB 8 = DISPLAY 9 = DISPLAYB 11 = TVO 12 = DSI 13 = VIC 14 = NVJPG 16 = DSIB 18 = NVDEC 19 = NVENC 20 = TSEC 21 = SOR 22 = SOR1 23 = DPAUX 24 = ISP 26 = ISPB 27 = VII2C
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
1	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB

Bit	Reset	Description
0	X	RWN: Read/write 0 = WRITE 1 = READ

14.10.19 NV_CLASS_HOST_INDDATA_0

These registers, when written, either write to the data to the INDOFFSET in INDOFF or trigger a read of the offset at INDOFFSET. This is an array of 31 identical register entries; the register fields below apply to each entry.

Offset: 0x2e..0x4c | Byte Offset: 0xb8..0x130 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INDDATA: Read or write data

14.10.20 NV_CLASS_HOST_LOAD_SYNCPT_PAYLOAD_32_0

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CHANNEL_SYNCPT_PAYLOAD

14.10.21 NV_CLASS_HOST_STALLCTRL_0

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0X0	ENABLE

14.10.22 NV_CLASS_HOST_WAIT_SYNCPT_32_0

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

14.10.23 NV_CLASS_HOST_WAIT_SYNCPT_BASE_32_0

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	BASE_INDX
7:0	X	INDX

14.10.24 NV_CLASS_HOST_LOAD_SYNCPT_BASE_32_0

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

14.10.25 NV_CLASS_HOST_INCR_SYNCPT_BASE_32_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

14.10.26 NV_CLASS_HOST_STALLCOUNT_HI_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	STALLCOUNT_HI

14.10.27 NV_CLASS_HOST_STALLCOUNT_LO_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	STALLCOUNT_LO

14.10.28 NV_CLASS_HOST_XFERCTRL_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0X0	ENABLE

14.10.29 NV_CLASS_HOST_CHANNEL_XFER_HI_0

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	CHANNEL_XFER_HI

14.10.30 NV_CLASS_HOST_CHANNEL_XFER_LO_0

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	CHANNEL_XFER_LO

14.11 Host Proto Channel Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

14.11.1 HOST1X_PROTCHANNEL_FIFOSTAT_0

CFNUMEMPTY is the number of free slots available in the per-channel command FIFO (needed for PIO or polling for completion of a wait).

Offset: 0x0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	INDRDY: Indicates that INDCOUNT==0, so it should be OK to issue another read
28:24	X	OUTFENTRIES: Number of entries available for reading in this channel's output FIFO
20:16	X	REGFNUMEMPTY: Register write/read FIFO free count
12	X	CFGATHER: Indicates whether or not GATHER is active. If a GATHER command issued via PIO, software must wait for the GATHER to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
11	X	CFEMPTY: Indicates whether the command FIFO is empty or not 0 = NOTEMPTY 1 = EMPTY
10:0	X	CFNUMEMPTY: Command FIFO free count

14.11.2 HOST1X_PROTCHANNEL_INDOFF_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
30	X	ACCTYPE: access type: indirect register or indirect frame buffer 0 = REG 1 = FB
29	X	BUF32B: Buffer up 32 bits of register data before sending it. Otherwise, register writes will be sent as soon as they are received. Does not support byte writes in 16-bit host. Does not affect frame buffer writes. 0 = NOBUF 1 = BUF
28:27	X	INDSWAP: Indirect frame buffer access swap control. 00 = No byte swap 01 = 16-bit byte swap ([31:0] -> {[23:16],[31:24],[7:0],[15:8]}) 10 = 32-bit byte swap ([31:0] -> {[7:0],[15:8],[23:16],[31:24]}) 11 = 32-bit word swap ([31:0] -> {[15:8],[7:0],[31:24],[23:16]}) 0 = NONE 1 = BYTE16 2 = BYTE32 3 = WORD32
25:18	X	INDMODID: ACCTYPE=REG: register module ID. This field should not be programmed to equal the module_id of Host1x. 0 = HOST1X 1 = DPAUX1 2 = VI 4 = TSECB 8 = DISPLAY 9 = DISPLAYB 12 = DSI 13 = VIC 14 = NVJPG 16 = DSIB 18 = NVDEC 19 = NVENC 20 = TSEC 21 = SOR 22 = SOR1 23 = DPAUX 24 = ISP 26 = ISPB 27 = VII2C
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
0	X	INDOFFUPD: Optionally disable the update of INDOFFSET when writing this register 0 = UPDATE 1 = NO_UPDATE

14.11.3 HOST1X_PROTCHANNEL_INDCNT_0

Indirect register access count

Used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. Only the protected channel should make liberal use of this feature for speeding up context switching.

For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0X0	INDCOUNT

14.11.4 HOST1X_PROTCHANNEL_INDDATA_0

This register, when written, writes the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again.

The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0X0	INDDATA: Read or write data

14.11.5 HOST1X_PROTCHANNEL_INDOFF2_0

Note: *This spec file contains additions to the "common" registers that cannot be put in the common section, otherwise all of the other registers will shift.*

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID2: ACCTYPE=REG: register module ID 0 = HOST1X 1 = DPAUX1 2 = VI 4 = TSECB 8 = DISPLAY 9 = DISPLAYB12 = DSI 13 = VIC 14 = NVJPG 16 = DSIB 18 = NVDEC 19 = NVENC 20 = TSEC 21 = SOR 22 = SOR1 23 = DPAUX 24 = ISP 26 = ISPB
31:2	X	INDOFFSET2: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET2: ACCTYPE=REG: register offset ([15:0])

14.11.6 HOST1X_PROTCHANNEL_TICKCOUNT_HI_0

This register holds the high 32 bits of the tick count value

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI

14.11.7 HOST1X_PROTCHANNEL_TICKCOUNT_LO_0

This register holds the low 32 bits of the tick count value

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO

14.11.8 HOST1X_PROTCHANNEL_CHANNELCTRL_0

This register will be used for controlling some channel-related commands including enabling/disabling of the tick counter and enabling/disabling of Kernel command filtering from the gather buffers.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0)

Bit	Reset	Description
2	DISABLE	KERNEL_FILTER_GBUFFER: Enable setclass command filter for gather buffers 0 = DISABLE 1 = ENABLE
0	DISABLE	ENABLETICKCNT: Enable or disable tick counter 0 = DISABLE 1 = ENABLE

14.11.9 HOST1X_PROTCHANNEL_PAYLOAD_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PAYLOAD

14.11.10 HOST1X_PROTCHANNEL_STALLCTRL_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_STALL

14.11.11 HOST1X_PROTCHANNEL_STALLCOUNT_HI_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_HI

14.11.12 HOST1X_PROTCHANNEL_STALLCOUNT_LO_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_LO

14.11.13 HOST1X_PROTCHANNEL_XFERCTRL_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_XFER

14.11.14 HOST1X_PROTCHANNEL_CHANNEL_XFER_HI_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_HI

14.11.15 HOST1X_PROTCHANNEL_CHANNEL_XFER_LO_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_LO

14.11.16 HOST1X_PROTCHANNEL_HOST1X_CHANNEL_SPARE_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:16	0xffff	CHANNEL_SPARE_HI
15:0	0x0	CHANNEL_SPARE_LO

14.11.17 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_HI_0

This register holds the high 32 bits of the tick count threshold value

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_THRESHOLD_HI

14.11.18 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_LO_0

This register holds the low 32 bits of the tick count threshold value

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_THRESHOLD_LO

14.11.19 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_CTRL_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	THRES_CMP: Enable or disable tick threshold compare 0 = DISABLE 1 = ENABLE

CHAPTER 15: VIDEO IMAGE COMPOSITOR (VIC)

The Video Image Compositor (VIC) unit implements video post-processing functions needed by a video playback application to produce the final image for the player window. The VIC can also perform scaling, composition, and rotation when no 3D rendering is involved.

The compositor implements much of the DirectX Video Acceleration 2.0 Enhanced Video Processor specification, including de-interlacing, scaling, color conversion, proc-amp, and compositing for up to 8 input surfaces. It supports advanced features like gamma/de-gamma programming, color correct processing, and pixel decompression.

15.1 Features

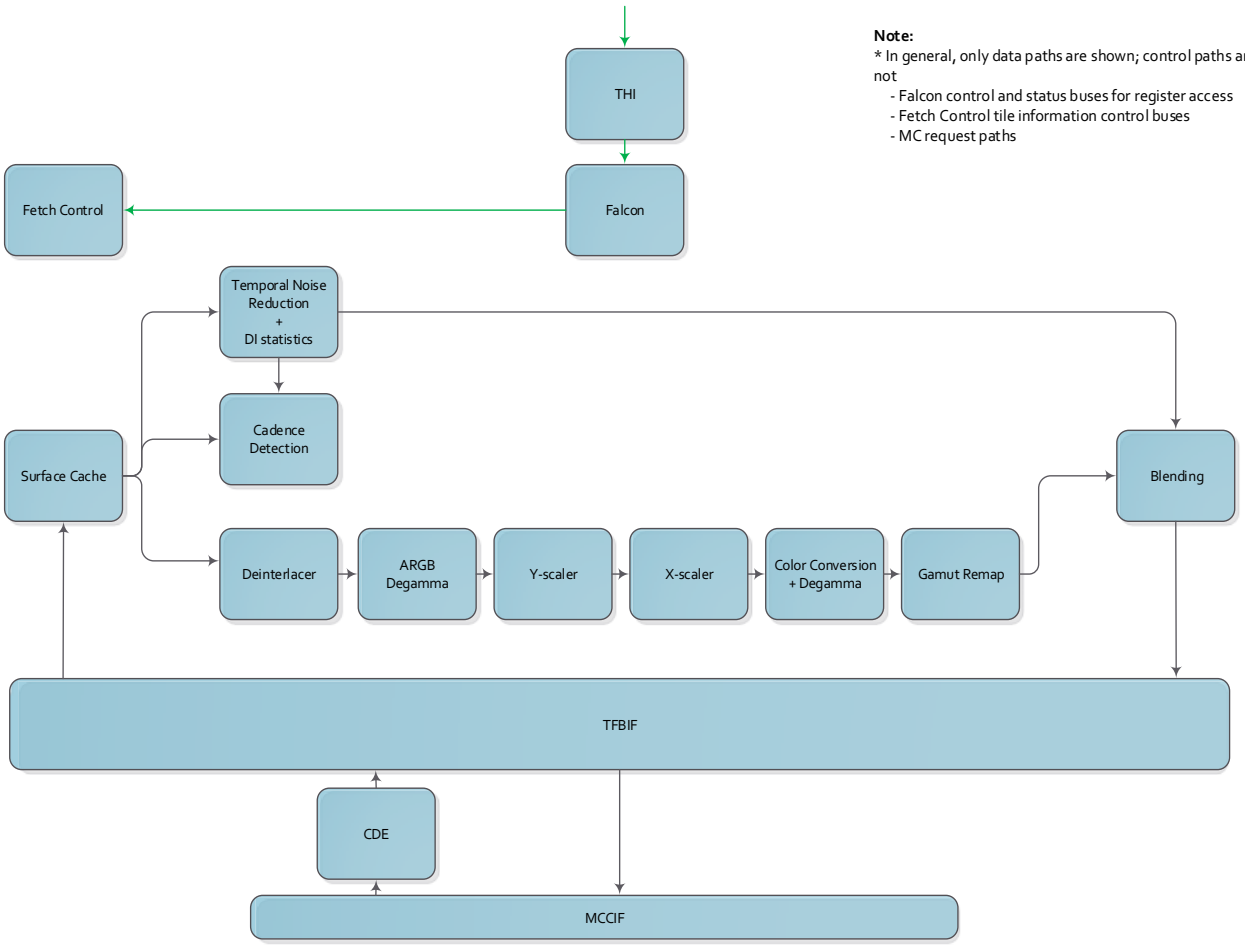
The VIC implements the following features:

- De-interlacing
- Inverse Teleciné
- Temporal Noise Reduction for Video and Camera
- Scaling (5-tap and 10-tap)
- Color and Pixel Format Conversion
- Memory Format Conversion
- Blend/Composite
- Rotation/Flips

15.2 Block Diagram Description

The following figure shows the top-level block diagram of the Video Image Compositor.

Figure 28: VIC Block Diagram



15.2.1 Falcon Microprocessor

The NVIDIA Falcon Microcontroller is used for method processing and performing processing related to inverse telecine.

15.2.2 Tegra Host Interface (THI)

The Tegra Host Interface (THI) shim handles incoming methods from the Host1x bus and passes those methods to the Falcon microcontroller, in addition to handling Tegra sync points.

15.2.3 Tegra Frame Buffer Interface (TFBIF)

The Tegra Frame Buffer Interface (TFBIF) interfaces with the Tegra X1 memory system. The TFBIF has one read and one write MCCIF client.

15.2.4 Color Decompression Engine (CDE)

The Color Decompression Engine reads compressed surfaces generated by the GPU.

15.2.5 Fetch Unit

During preprocessing (IVTC passes) the fetch control breaks the output surfaces into 16x16 byte tiles and processes columns of these tiles at a time.

During main processing, the fetch control breaks the output surface up into 64x16 pixel tiles and processes those individually. For each tile, the fetch unit loops over all input surfaces.

When processing each tile, 64*16+*n* pixels need to be fetched, where *n* is the extra padding needed for scaling and deinterlacing. The correct input tile size to be fetched is (64*s_x+n)x(16*s_y+n), where s_x and s_y are the horizontal and vertical downscale factors. "n" here depends on the filter taps specified (1, 2, 5 or 10-tap), as well as the deinterlacing algorithm used.

Each tile being fetched is stored in the Surface Cache. This cache holds 128 entries of 256 bytes.

15.2.6 Inverse Telecine (IVTC) Unit

The IVTC unit runs on individual video streams at a time and consists of 2 separate passes.

The first pass does noise reduction and motion adaptation buffer updates. It also calculates the weave and artifact counts used for the IVTC decision. The final decision for IVTC is done in Falcon microcode.

The second pass combines the updated motion adaptation buffer and the previous motion buffer (which has a different parity) and combines them for DiSi1 deinterlacing during the main processing pass.

15.2.7 Scale Unit

The Scale Unit consists of three blocks:

- The first block deinterlaces the input. The deinterlacing modes supported are DiSi1, BOB, and WEAVE. Weave is only used on progressive content or when inverse telecine detected a cadence.
- The second block scales the data in the vertical direction producing at most 16 lines of output to the next stage.
- The third block scales the data in the horizontal direction producing at most 64 pixels of output.

15.2.8 Blend Unit

The Blend Unit consists of four blocks:

- The first block buffers the incoming data and converts from any input order into interleaved pixel order (either ARGB or AYUV).
- The second block performs all color conversion tasks.
- The third block blends data into the output buffer. This stage is also used during preprocessing.
- The fourth block is an input-output DMA that can read original buffer data from the memory system and write back the final blended result. The reading is only used during preprocessing.

15.3 Functionality

15.3.1 Configurability

The VIC operates at 8 pixels/clock. It supports 8 slots.

15.3.2 Color Correct Processing

When processing colors, the VIC supports the following:

- Full sRGB support
- Precise internal pixel format precision that:
  - satisfies the DirectX srgb2linear and linear2srgb error requirements
  - is visually indistinguishable from the GPU output
- Linear Space Scaling and Blending

15.3.3 Config Structure

The Config Structure is the primary programming interface that allows software to talk to the VIC unit. The Config Structure is a series of sub-structures which define the full use case to be executed by VIC. Software sets up the ConfigStruct in DRAM; the VIC hardware then reads in this structure from memory and interprets it to decide on the operations that need to be performed.

15.3.4 Pixel Decompression

VIC can decompress pixels compressed by the GPU ROP unit. Refer also to Chapter 25: Color Decompression Engine in this document.

The supported formats, decompression modes, and surface sizes are:

- Pixels formats supported: 32-bit per pixel formats such as A8R8G8B8; only single plane formats are supported
- Memory formats supported: Block-linear 2CRA compressed surfaces
- Supported decompression modes are:
  - Arithmetic decompression (1:2 decompression)
  - Reduction compression (1:8 decompression)
  - ZBC decompression for solid color tiles.
- Slot support: Decompression is supported on all 8 slots. Software has to indicate through the config structure if compression is enabled for each slot.
- The maximum surface size supported for any compressed surface is 4096x4096 pixels

The Color Decompression Engine (CDE) transparently compresses regular requests sent by the VIC unit, and decompresses data in the read responses on-the-fly before sending the data back to the client. To decompress the compressed pixel data that is fetched from memory, the CDE needs to store the compbit information for every ROP tile (32Bx8) that is fetched from memory in a local buffer known as the CompTagBuffer (CTB). This local buffer is maintained as a shared resource for all 8 slots that need to be partitioned among the slots at the beginning of the frame. This partitioning is done in the Falcon microcode.

A worst-case use case of 2:1 downscaling on all 8 slots is supported. If certain slots are not being scaled, it is possible to re-allocate the CTB entries to the other slots instead so that some slots can support a larger downscale at the expense of the non-scaled slots CTB entries.

The following new methods have been added to the VIC class to let software set the comp-tag buffer base addresses - SlotNCompressBitOffset(), and the expected method data is bits 39:8 of the buffer base, just like all the other Offset() methods.

The following fields have been added to the VIC config structure for decompression support:

- DecompressEnable – 1 bit (per slot) in the SlotConfig structure
- DecompressCtbCount – 8 bits (per slot) in the SlotConfig structure. This field is used if the driver implements the CTB partitioning code listed above instead of the Falcon microcode.
- DecompressZbcColor – 32 bits (per slot) in the SlotConfig structure

Refer to Chapter 24: Display Controller in this TRM for more information on the CDE.

15.3.5 Slots

The VIC supports up to 8 different input pictures that can be composited on to a single surface. Support for 8 input pictures enables composition of multiple streams in a single pass. For example, for Blu-Ray content:

- Background
- Primary Video
- Secondary Video
- Presentation Graphics

- Interactive Graphics

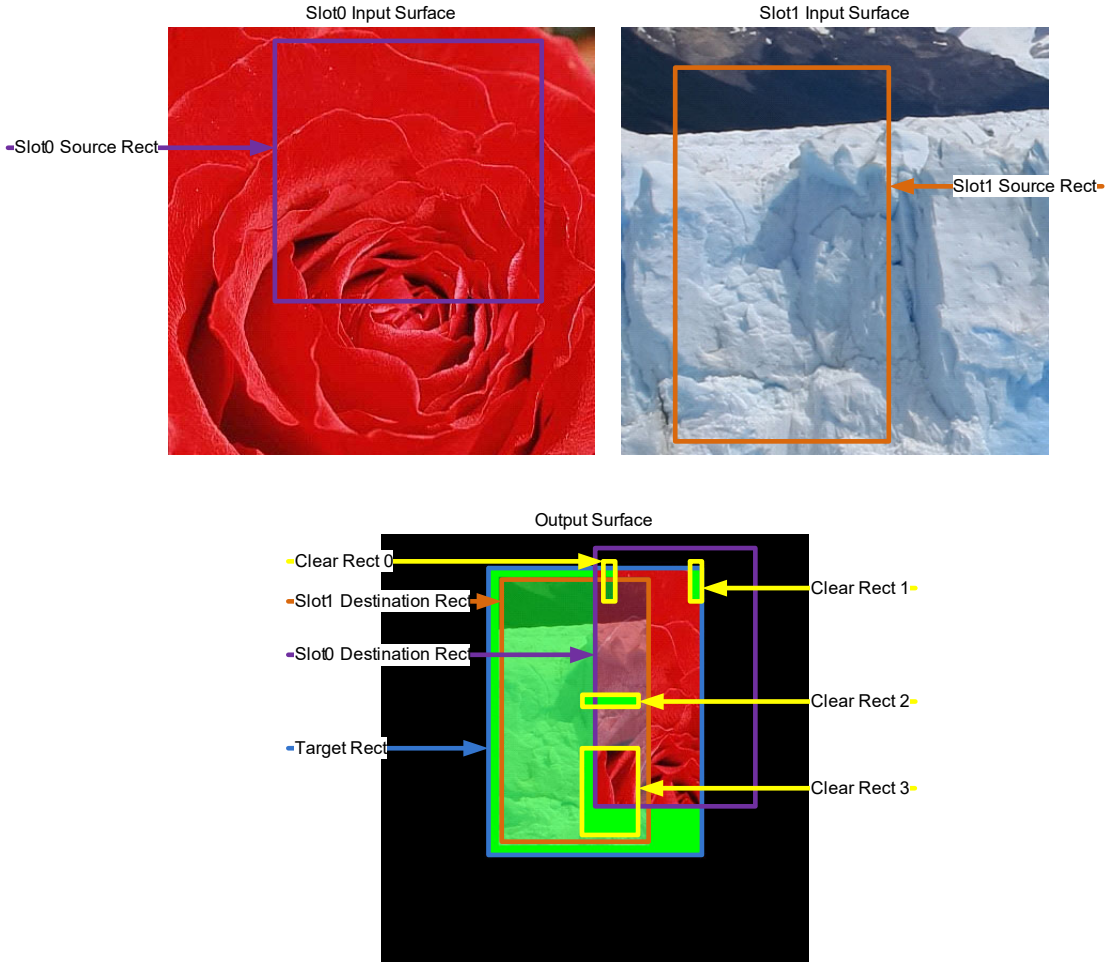
Each slot has a state that is specified through the config structure. Control parameters, such as the input formats, deinterlacing, scaling factors, clipping, color-space conversion, are independently set per input surface. All slots support the same functionality.

During the composition stage, the slots are blended in ascending order (that is, the output of slot1 will be blended onto slot 0, the output of slot 2 is blended on to slot 1, etc.).

Each slot can be programmed to access multiple surfaces. The number of surfaces used depends on the type of slot input (e.g., interlaced versus progressive), input format (e.g., packed versus planar color formats), and also on the type of processing (e.g., temporal noise reduction versus cadence detection versus deinterlacing).

15.3.5.1 Surface Composition

Figure 29: Programming Input and Output Surface Parameters



The maximum dimension of any image on the input or output sides of the VIC is 16384 pixels. The dimensions of each input slot surface can be defined along with the pixel and memory format types in the VIC Config Structure.

A source rectangle defines the region of pixels of the input slot surface that will contribute to the composition of the output surface, and a destination rectangle defines the region of the output surface that is affected by a given input slot. Together, the source and destination rectangle parameters specify the scaling ratios desired. Each slot can also dictate how pixel data is laid out in each surface cache entry by specifying the amount of pixel data, i.e., the number of bytes wide, logically represented by each cache entry; this provides flexibility in reducing the memory over-fetch associated with fetch of surfaces under various use cases (e.g., scaling ratios, deinterlacing modes, etc.).Via the OutputConfig structure, the dimensions of the output surface can be defined along with the pixel and memory format types. The same structure also defines a target rectangle to restrict the pixel

processing output to a certain rectangle in the output surface. A programmable color can also be set to fill the background of the target rectangle.

The VIC allows the programming of clear rectangles that prevent the fetch of pixels from specific rectangular regions of input slots. The use of the clear rectangles can be used to reduce redundant pixel fetches, for example, if an opaque surface is known to occlude a layer below it, a clear rectangle can be specified to prevent fetching of pixels from the occluded region of the lower layer. Up to eight clear rectangles are specified via the ClearRectStruct structs, and per-slot clear rectangle mask enables are specified via the SlotConfig struct, Clear rectangles are specified relative to the output surface co-ordinate system.

With the addition of the sub-pixel source rectangle feature, source rectangle coordinates are specified in the config structure in a U14.16 format, allowing the source rectangle to be specified at a sub-pixel resolution. Destination, target, and clear rectangles coordinates are pixel aligned.

15.3.6 Memory Format Support

The following memory formats must be supported on input and output. Input and output memory formats do not have to match.

- Pitch Linear
- Block Linear (16Bx2 kind with sector ordering)
  - 64x8 byte GOB format with 16x2 byte sectors
  - Contiguous 64B atoms are laid out as 32x2 byte blocks

The Tegra memory controller is optimized for 64B requests, and all VIC requests should be optimized to issue at least 64B sized requests.

The VIC Surface Cache supports the above memory formats on input when fetching pixel data from memory.

The VIC Output Buffer supports the above memory formats on input when reading background image data when pre-processing inverse telecine data from memory, and on the output side when writing composited image data back to memory.

Memory formats supported by the VIC are enumerated as follows:

Table 48: Memory Format Enumerations

Surface Kind Enumeration	Encoding (Four Bits)
BLK_KIND_PITCH	0x0
BLK_KIND_GENERIC_16Bx2_TEGRA	0x1

To fully support the Block Linear format, block height is programmable to any one of the following enumerations.

Table 49: Block Height Enumerations

Block Height Enumeration	Value	Encoding (Four Bits log2 Encoding)
ONE_GOB	1	0
TWO_GOBS	2	1
FOUR_GOBS	4	2
EIGHT_GOBS	8	3
SIXTEEN_GOBS	16	4
THIRTYTWO_GOBS	32	5

15.3.6.1 Surface Transformation

The Blend unit is responsible for generating the destination surface addressing that will implement mirroring (across X-axis and/or Y-axis) of tiles within the Target rectangle of the output surface.

The output buffer can mirror pixels in the X and Y directions within each output tile. This ability in addition to the fetch control changes mentioned above gives the ability to flip the contents of the entire Target rectangle about its axis.

The output buffer also supports transposition of the Target Rectangle. This ability when combined with target rectangle flips gives the ability to rotate images by any multiple of 90 degrees.

Rotation must be done in the pipeline after deinterlacing because the deinterlacer assumes a normal orientation of the input fields (interlaced field lines run horizontal). Rotation of the image prior to deinterlacing will make the deinterlacer ineffective.

Figure 30: Interlaced Field Lines Run Horizontally

The programmed starting address of the source and destination rectangles will remain at the upper left corner of the unit. The ‘Source Image’ diagrammed below shows the logical tile traversal pattern as the input rectangle traverses the VIC unit (tiles are read in raster order from left to right, then top to bottom), and the destination rectangle diagrammed below shows the expected transformations when the programmer specifies the combination of flipX, flipY, and transposeXY transformations available to be applied to the output rectangle.

Figure 31: Surface Transformation Orientations

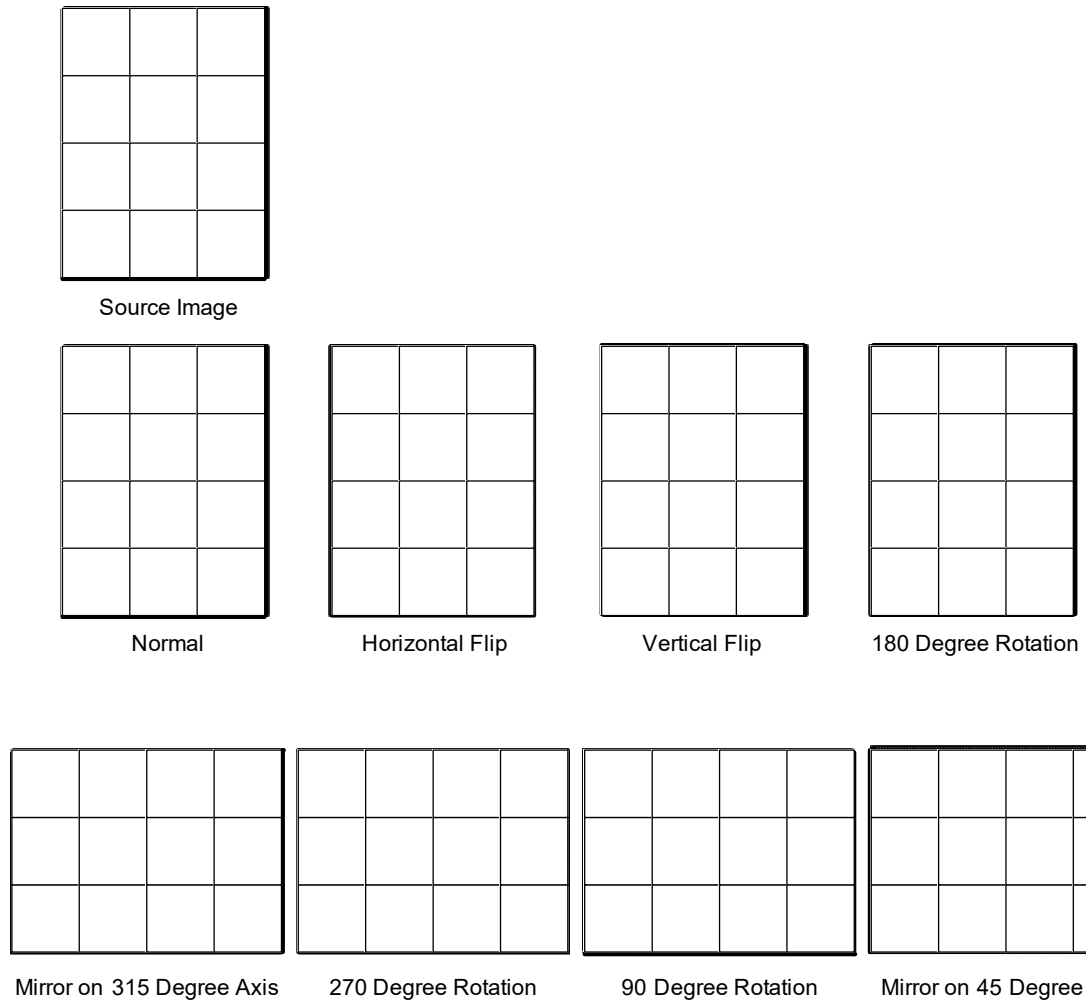


Table 50: Output Buffer Surface Transformation Controls

Transformation	transposeXY	flipY	flipX
Normal (no transformation)	0	0	0
H flip (mirror on vertical axis)	0	0	1
V flip (mirror on horizontal axis)	0	1	0
180-degree rotation	0	1	1
Mirror on 315-degree axis	1	0	0
270-degree rotation	1	1	0
90-degree rotation	1	0	1
Mirror on 45-degree axis	1	1	1

Surface mirroring (FlipX/FlipY) is accomplished via programming of the OutputConfig structure. Surface transposition is accomplished via programming of the OutputTranspose field of the OutputConfig structure structure.

The table below shows the output image for a certain surface composition and how the output transformations affect the final output. The results of the transformations are summarized in the rules below.

1. The output surface parameters (SurfaceWidth/Height, LumaWidth/Height, and ChromaWidth/Height) are transposed when output transpose is enabled. This means that if the target rectangle/destination rectangles fit within the output surface before transformation, they will also fit within the output surface after all transformations are done.
2. The position of the target rectangle in the output surface gets flipped or transposed along with all of its contents (clear rectangles and destination rectangles) when OutputFlipX/OutputFlipY/OutputTranspose are enabled.

3. When output flips are enabled, the entire contents of the target rectangle are flipped, including all ClearRects and DestinationRects and their contents. The axis of the flip is the axis of the output surface.
4. When OutputTranspose is enabled, the position of the target rectangle is transposed, as well as all of its contents (including all clear/destination rectangles).
5. When transpose and flips are enabled together, the overall effect of the transformation happens as if the flip operations occur first, followed by the transpose operation.

15.3.7 Memory Copy

The VIC memory copy bandwidth scales with the VIC core frequency. The VIC core pixel processing pipeline can process 2, 4, 8, or 16 pixels per VIC clock. The Tegra X1 VIC unit is sized to operate on 8 pixels per clock.

15.3.8 Temporal Noise Reduction

VIC 1.0 implemented a temporal noise reduction algorithm based on a motion adaptive IIR filter, where the filter weight was adapted based on a few constants and the SOS difference between a 3x3 neighborhood of pixels around the current pixel in the current and previous frames. This same algorithm is supported in VIC 3.0 and 4.0 as well.

The VIC 1.0 TNR algorithm (hereafter called TNR1) works well for videos or captures that have good lighting conditions and lower levels of noise in the input video. However, when the lighting conditions are poor and there is a lot of noise in the input video/capture, the algorithm tries to be safe by not modifying it much, and allows most of the noise to pass through into the output.

The TNR2 algorithm performs significantly better in the above mentioned cases by pre-filtering the data spatially and allowing significantly higher alpha values in the IIR blend. This algorithm can be turned on by programming the AdvancedDenoise* fields in the SlotConfig structure. The TNR2 algorithm also requires information about the lighting conditions at which the video or capture is being shot (information that will probably be derived from ISP statistics), and this information should be programmed into the LightLevel fields in the SlotConfig structure (4 bits, 0-darkest, 15-brightest).

15.3.9 Inverse Telecine

There are two statistics that the VIC hardware calculates: the difference count and the artifact count. The VIC hardware just calculates the statistics; a state machine on the Falcon Microcontroller has to interpret these.

15.3.10 Deinterlacing

The VIC supports various modes of deinterlacing the input video content such as DiSi1, BOB, and WEAVE. Weave is only used on progressive content or in case inverse telecine detected a cadence.

15.3.11 Color Fill

Programming the output surface with only a background color and writing to memory will produce a color fill.

15.3.12 Pixel Format Support

The VIC pipeline supports an increased number of pixel formats compared to what is currently comprehended by VIC1.0 on input and output. The VIC pipeline can accept a surface of a given input pixel format and produce a given output pixel format.

The table below lists the supported pixel formats. The symbols in color conversion mode column describe the internal VIC representation of the format, (e.g., 1P_1C represents 1 plane with 1 component pixels, 2P_1+2C represents 2 planes, 1 plane with 1 component pixels and a second plane with 2 component pixels).

Table 51: Pixel Format Support

Pixel Format	FOURCC	Input Support	Output Support	Rotation	Scaling	High Quality Deinterlacing	New index	Color_Conversion_Mode
T_A8		Y	Y	Y	Y	N	0	1P_1C
T_L8/ T_Y8/ T_YUV100		Y	Y	Y	Y	N	1	1P_1C

Table 51: Pixel Format Support

Pixel Format	FOURCC	Input Support	Output Support	Rotation	Scaling	High Quality Deinterlacing	New index	Color_Conversion_Mode
T_A4L4		Y	Y	Y	Y	N	2	1P_2C
T_L4A4		Y	Y	Y	Y	N	3	1P_2C
T_R8		Y	Y	Y	Y	N	4	1P_1C
T_A8L8		Y	Y	Y	Y	N	5	1P_2C
T_L8A8		Y	Y	Y	Y	N	6	1P_2C
T_R8G8		Y	Y	Y	Y	N	7	1P_2C
T_G8R8		Y	Y	Y	Y	N	8	1P_2C
T_B5G6R5		Y	Y	Y	Y	N	9	1P_3C
T_R5G6B5		Y	Y	Y	Y	N	10	1P_3C
T_B6G5R5		Y	Y	Y	Y	N	11	1P_3C
T_R5G5B6		Y	Y	Y	Y	N	12	1P_3C
T_A1B5G5R5		Y	Y	Y	Y	N	13	1P_4C
T_A1R5G5B5		Y	Y	Y	Y	N	14	1P_4C
T_B5G5R5A1		Y	Y	Y	Y	N	15	1P_4C
T_R5G5B5A1		Y	Y	Y	Y	N	16	1P_4C
T_A5B5G5R1		Y	Y	Y	Y	N	17	1P_4C
T_A5R1G5B5		Y	Y	Y	Y	N	18	1P_4C
T_B5G5R1A5		Y	Y	Y	Y	N	19	1P_4C
T_R1G5B5A5		Y	Y	Y	Y	N	20	1P_4C
T_X1B5G5R5		Y	Y	Y	Y	N	21	1P_4C
T_X1R5G5B5		Y	Y	Y	Y	N	22	1P_4C
T_B5G5R5X1		Y	Y	Y	Y	N	23	1P_4C
T_R5G5B5X1		Y	Y	Y	Y	N	24	1P_4C
T_A4B4G4R4		Y	Y	Y	Y	N	25	1P_4C
T_A4R4G4B4		Y	Y	Y	Y	N	26	1P_4C
T_B4G4R4A4		Y	Y	Y	Y	N	27	1P_4C
T_R4G4B4A4		Y	Y	Y	Y	N	28	1P_4C
T_A8B8G8R8		Y	Y	Y	Y	N	29	1P_4C
T_A8R8G8B8		Y	Y	Y	Y	N	30	1P_4C
T_B8G8R8A8		Y	Y	Y	Y	N	31	1P_4C
T_R8G8B8A8		Y	Y	Y	Y	N	32	1P_4C
T_X8B8G8R8		Y	Y	Y	Y	N	33	1P_4C
T_X8R8G8B8		Y	Y	Y	Y	N	34	1P_4C
T_B8G8R8X8		Y	Y	Y	Y	N	35	1P_4C
T_R8G8B8X8		Y	Y	Y	Y	N	36	1P_4C
T_A2B10G10R10		Y	N	Y	Y	N	37	1P_4C
T_A2R10G10B10		Y	N	Y	Y	N	38	1P_4C
T_B10G10R10A2		Y	N	Y	Y	N	39	1P_4C
T_R10G10B10A2		Y	N	Y	Y	N	40	1P_4C
T_A4P4		Y	N	Y	Y	N	41	1P_2C
T_P4A4		Y	N	Y	Y	N	42	1P_2C
T_P8A8		Y	N	Y	Y	N	43	1P_2C
T_A8P8		Y	N	Y	Y	N	44	1P_2C
T_P8		Y	N	Y	Y	N	45	1P_1C
T_P1		Y	N	Y	Y	N	46	1P_1C
T_U8V8		Y	Y	Y	Y	N	47	1P_2C

Table 51: Pixel Format Support

Pixel Format	FOURCC	Input Support	Output Support	Rotation	Scaling	High Quality Deinterlacing	New index	Color_Conversion_Mode
T_V8U8		Y	Y	Y	Y	N	48	1P_2C
T_A8Y8U8V8 (AYUV444 packed)		Y	Y	Y	Y	N	49	1P_4C
T_V8U8Y8A8 (AYUV444 packed)		Y	Y	Y	Y	N	50	1P_4C
T_Y8_U8__Y8_V8 (YUV422 packed)	YUY2/YUYV	Y	Y	Y	Y	Y	51	1P_4C
T_Y8_V8__Y8_U8 (YUV422 packed)	YVYU	Y	Y	Y	Y	N	52	1P_4C
T_U8_Y8__V8_Y8 (YUV422 packed)	UYVY	Y	Y	Y	Y	Y	53	1P_4C
T_V8_Y8__U8_Y8 (YUV422 packed)		Y	Y	Y	Y	N	54	1P_4C
T_Y8___U8V8_N444 (YUV444 semi-planar)		Y	Y	Y	Y	N	55	2P_1+2C
T_Y8___V8U8_N444 (YUV444 semi-planar)		Y	Y	Y	Y	N	56	2P_1+2C
T_Y8___U8V8_N422 (YUV422 semi-planar)	e.g., NV61	Y	Y	Y	Y	N	57	2P_1+2C
T_Y8___V8U8_N422 (YUV422 semi-planar)	e.g., NV16	Y	Y	Y	Y	N	58	2P_1+2C
T_Y8___U8V8_N422R (YUV422R semi-planar)		Y	Y	Y	Y	N	59	2P_1+2C
T_Y8___V8U8_N422R (YUV422R semi-planar)		Y	Y	Y	Y	N	60	2P_1+2C
T_Y8___U8V8_N420 (YUV420 semi-planar)	e.g., NV21	Y	Y	Y	Y	N	61	2P_1+2C
T_Y8___V8U8_N420 (YUV420 semi-planar)	e.g., NV12	Y	Y	Y	Y	Y*	62	2P_1+2C
T_Y8___U8___V8_N444 (YUV444 planar)**	e.g., YV24	Y	Y	Y	Y	N	63	3P_1+1+1C
T_Y8___U8___V8_N422 (YUV422 planar)**	e.g., YV16	Y	Y	Y	Y	N	64	3P_1+1+1C
T_Y8___U8___V8_N422R (YUV422R planar)**		Y	Y	Y	Y	N	65	3P_1+1+1C
T_Y8___U8___V8_N420 (YUV420 planar)**	e.g., YV12	Y	Y	Y	Y	Y	66	3P_1+1+1C

* DiSi1 deinterlacing only works on NV24 surfaces and field-based variants of NV12, YV12, YUY2, and UYVY formats (with each field stored in a separate surface).

** VIC supports three independent surface pointers for each of the input slots and output surfaces, one each for luma, chroma-u and chroma-v.

In the color format, the notation called “A:B:C” notation is used to describe how often U and V are sampled relative to Y.

- 444 (4:4:4) means no downsampling of the chroma channels.
- 422 (4:2:2) means 2:1 horizontal downsampling, with no vertical downsampling. Every scan line contains four Y samples for every two U and V samples.
- 420 (4:2:0) means 2:1 horizontal downsampling, with 2:1 vertical downsampling.
- R refers to ‘rotated’. 422 by default means horizontal downsampling; while 422R means vertical downsampling. Instead of the chroma being shared between 2 horizontally adjacent luma components, it is shared between 2 vertically adjacent luma components.

The VIC does not support 24-bit color formats, either RGB or YUV.

15.3.13 Scaling and Filtering

The VIC contains Y and X-scaler units that together allow 5- or 10-tap polyphase filtering and scaling of the input surface. The hardware can be programmed with a set of filter coefficients for each filter-type (Detail/Noise/Normal), phase (0,1/32,2/32, .. 32/32), scale ratio (1:1, 2:1, 4:1, 8:1, 16:1), filter_length (5-tap, 10-tap) and stream (sub-picture or normal). Based on the actual scale-ratio, phase, etc., the hardware can then interpolate between these programmed values for use in the actual filters.

There are four filter types (5-tap non-substream, 5-tap substream, 10-tap non-substream, 10-tap substream). Five tables are needed for each coefficient type as different tables are needed for different scale down ratio of 1:1, 2:1, 4:1, 8:1 and 16:1[1]. Thus 20 tables are needed with a maximum downscale ratio of 16:1.

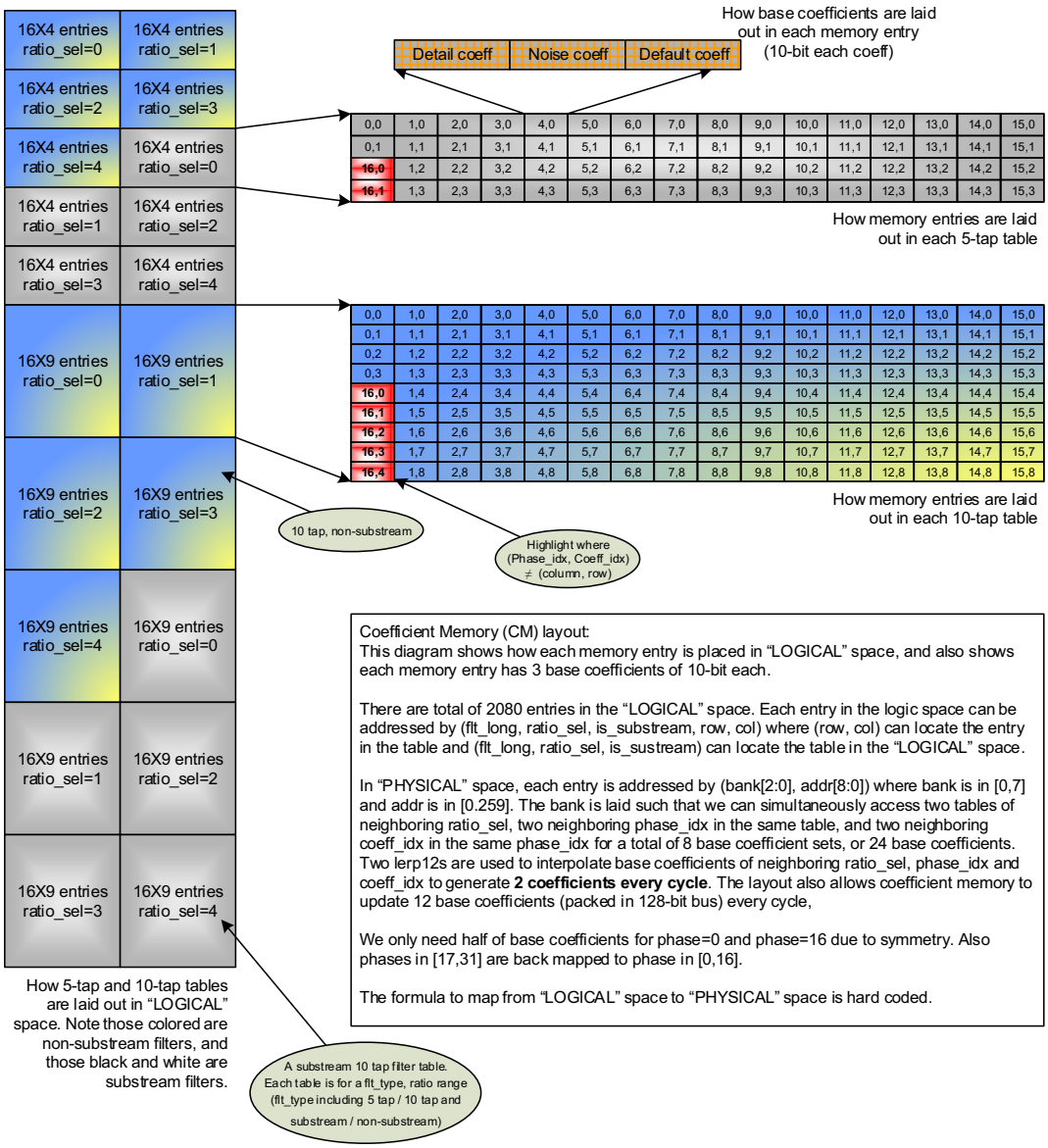
Each table has 16X4 memory entries (for the 5-tap filter) or 16X9 memory entries (for 10-tap filter). Each memory entry has three 10-bit base coefficients (for Detail/Noise/Default) that will be feed into a LERP3 simultaneously to interpolate based on different noise and detail weight. The table has 16 columns, and each column stores all the coefficients for phase 0/32, 1/32, 2/32, 3/32, 4/32, 5/32, 6/32, 7/32, 8/32, 9/32, 10/32, 11/32, 12/32, 13/32, 14/32 and 15/32, with the exception that the first column stores coefficients for both 0/32 and 16/32.

Each column stores four sets of base coefficients for the 5-tap filter because the last coefficient can be derived from the remaining four. phase 0/32 and 16/32 are special cases in that only two sets of base coefficients are needed for each of them in the 5-tap case,only four sets of base coefficients are needed for phase 0/32 in the 10-tap case, and five sets of base coefficients are needed for phase 16/32 in the 10-tap case. Therefore, phase 0/32 and 16/32 are folded into the same column. Due to symmetry, the base coefficients for phases 17 through 31 do not need to be stored.

The VIC also implements 1-tap (nearest-neighbor) and 2-tap (bi-linear) filtering modes. These modes can be used when power and bandwidth are of greater concern than quality. The 1-tap and 2-tap modes use filter coefficients that are based only on the output pixel phase with respect to the input pixel grid, and do not use the programmable coefficient tables at all. As a result, Detail Filtering or Noise Filtering can be disabled when using these modes.

1. 1:1 ratio contains identify filter that can be used for all scale up cases. Different scale down ratio requires different coefficients since the filter is a low pass filter the bandwidth of which is related to the scale ratio. Coefficients for 2:1, 4:1, 8:1 and 16:1 are stored to interpolate any downscale ratios.

Figure 32: Filter Coefficient Arrangement



15.3.13.1 Filter Override Mode

The VIC unit supports a filter override mode that can be enabled by setting both DownsampleHoriz and DownsampleVert in the config structure to 0. In this mode, software can specify the exact filter kernels that it wants to apply independently for each slot, and again independently for luma and chroma planes. The filter kernels thus specified are directly applied to the pixels, with the hardware skipping the interpolations for detail/noise/scaling and for the scale ratio. The only hardware interpolation done in this mode is between adjacent phase values based on the actual pixel phase.

Because of the restricted size of the FilterCoefficient table, there is only enough space to specify coefficients for 5 slots; thus only the first 5 slots can be enabled when the filter coefficient override mode is enabled.

15.3.13.2 Panoramic Scaling

The VIC also implements a "panoramic" scaling mode which can be used to convert images between different aspect ratios while avoiding any letterboxing. This mode essentially implements a non-linear scaling in the horizontal direction, with the scale ratio changing as we move away from the center of the image.

Panoramic scaling has an additional parameter *p* ([-1, 1]) that defines the magnitude of the second order term in a polynomial. The polynomial parameters for scaling can then be defined as follows.

The original scaling factor *f* is changed based on the location *x* ([-1, 1]) within the destination rectangle.

$$scaling\_factor_x = f * (a + 3bx^2) \quad \text{with} \quad a = \begin{cases} 1 - \frac{1}{2}p & p < 0 \\ 1 - p & p \geq 0 \end{cases} \quad \text{and} \quad b = \begin{cases} \frac{1}{2}p & p < 0 \\ p & p \geq 0 \end{cases}$$

The source sample position *x'* is therefore calculated as:

$$x' = f * (ax + bx^3) = a'x + b'x^3 \quad \text{with} \quad a' = fa \quad \text{and} \quad b' = fb$$

The new scaling factor can be calculated incrementally using 2 counters.

Panoramic scaling for 4:3 to 16:9 conversions should have $p = -\frac{2}{3}$, while 16:9 to 4:3 conversions should have $p = \frac{1}{4}$ for 1:1 scaling in the center.

15.3.13.3 Detail Filter Clamping

When using the Detail Filtering mode for edge enhancement, etc., it is possible for large pixel overshoots that result in a "ghosting" effect in the filtered output. To avoid such large overshoots, the Detail Filter clamping mode prevents the output pixel from being very different from the center input pixel of the filter support region.

For example, if the DetailFltClamp value in the SlotConfig structure is set to 10, and we are doing 5-tap filtering, then the output pixel value in each of the scalers (Y/X) is clamped so that it is within +/- 10 of the center (third) input pixel of the 5-tap filter support. Setting the DetailFltClamp value to 0 disables the clamping.

15.3.14 Pixel Format Conversion

Color conversion and proc-amp are implemented using a matrix multiplication on every pixel. The matrix only changes when the surfaces changes.

Current chain of operations:

- Proc-amp + Output CC (4x3 matrix)
- Clamping (optional soft clamping, 2 control entry)

The 4x3 matrix values are specified in the MatrixStruct as S12.8 values. In addition, the result of the matrix multiplication will be right shifted by r_shift. To allow for highest accuracy r_shift should always be as high as possible without losing any range in the other coefficients. Note that the constant offsets (c03, c13, c23) are S12.8 and are not affected by r_shift.

$$\begin{pmatrix} out0 \\ out1 \\ out2 \end{pmatrix} = \begin{pmatrix} c00 & c01 & c02 & c03 \\ c10 & c11 & c12 & c13 \\ c20 & c21 & c22 & c23 \end{pmatrix} \begin{pmatrix} in0 \gg r\_shift \\ in1 \gg r\_shift \\ in2 \gg r\_shift \\ 1 \end{pmatrix}$$

Soft Clamping

Soft clamping defines a piecewise linear function consisting of three pieces. The control entries define the area of soft clamping. For example, the following equation applies for the values *a* and *b*:

$$v' = \begin{cases} 0 & v < -a \\ \frac{v+a}{2} & -a \le v < a \\ v & a \le v < b \\ \frac{v+b}{2} & b \le v < 2-b \\ 1 & 2-b \le v \end{cases}$$

When converting between pixel formats with components of varying bit width, bit replication is used for expanding component bit widths, and truncation is used for reducing component bit widths.

Dithering and undithering are not supported.

Handling Subsampled Chroma Formats

The chroma location for pixel formats with subsampled chroma can be specified using the ChromaLocHoriz and ChromaLocVert parameters in the VIC Config structure. Each of these parameters can take values 0, 1, or 2 representing a chroma location as in the below table.

Table 52: Chroma Location

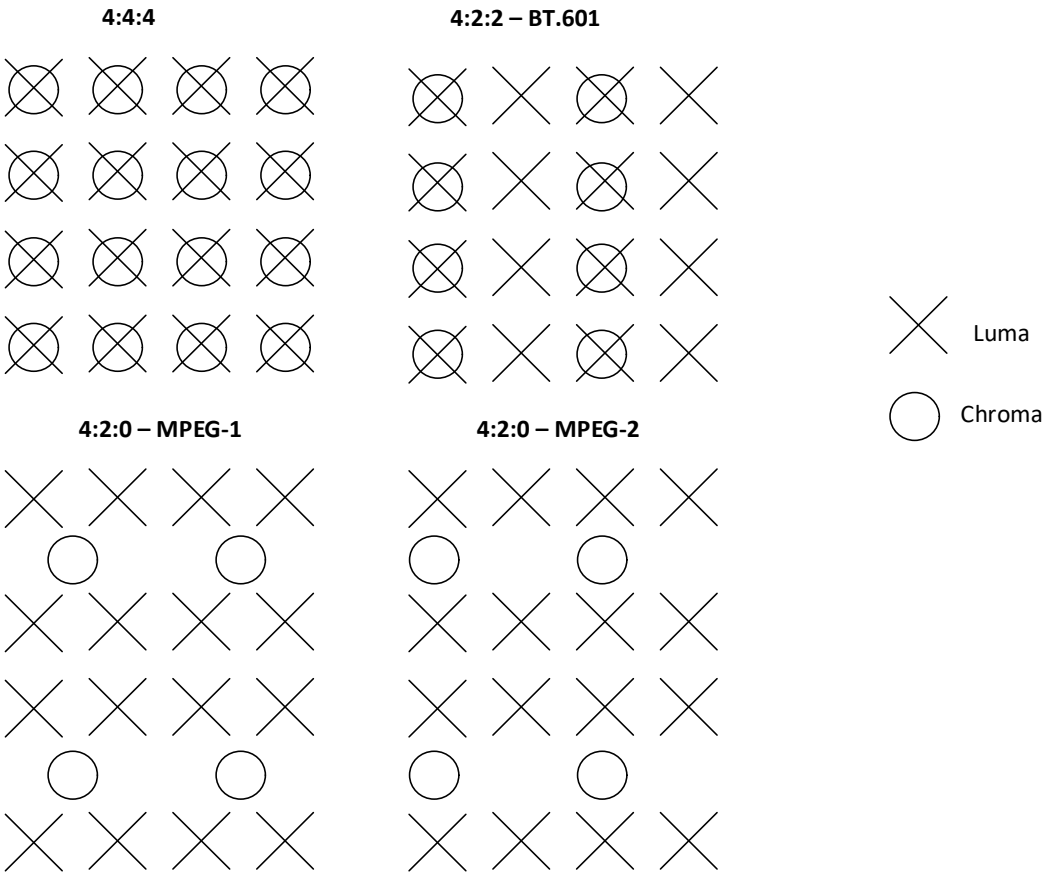
ChromaLoc	Chroma Position
0	Co-sited with even luma values
1	Midway between even and odd luma values
2	Co-sited with odd luma values

For example, the following table shows the ChromaLocHoriz and ChromaLocVert parameters for four YUV formats shown in the figure below.

Table 53: Example ChromaLoc Values

Format	ChromaLocHoriz	ChromaLocVert	Comment
4:4:4	Don’t Care	Don’t Care	Not subsampled
4:2:2 – BT601	0	Don’t Care	
4:2:0 – MPEG1	1	1	
4:2:0 – MPEG2	0	1	

Figure 33: Example YUV Formats



Data Size Conversion

When expanding the number of bits in a component (e.g., from RGB565 to RGB888), expansion should use bit replication of the MSBs of the original value to fill in the LSBs of the expanded bit length value. For example, if converting from a 5-bit value to an 8-bit value, the first 3 MSBs of the source 5-bit value should be concatenated as the 3 new LSBs to the original 5-bit value to produce the new expanded 8-bit value.

For example: 5-bit R5=11000’b expanded to 8 bit R8=11000110’b

When reducing the number of bits in a component (for example, from RGB888 to RGB565), the LSBs of the old value should be truncated to arrive at the shortened value.

For example: 8-bit R8=11000000’b reduced to 5-bit R5=11000’b

Replication followed by truncation will ensure that the truncated value will equal the original pre-replicated operand, i.e., RGB565 >RGB888 expansion followed immediately by RGB888 > RGB565 reduction will result in the same original value.

Luminance Conversion

Conversion from luminance-only color formats (e.g., L8/Y8/YUV100, A8L8) to and from RGB or YUV formats should be done as outlined below.

- Luminance-only to YUV

  L > Y, 128 > U, 128 > V
- Luminance-only to RGB

  L > R, L > G, L > B
- YUV to Luminance-only

  Y > L

- Clamp L between 0.0 and 1.0
- RGB to Luminance-only

  (R*5 + G*9 + B*2)/16 > L
- Clamp L between 0.0 and 1.0

Adding and Removing Alpha

If converting from a pixel format with alpha to a pixel format with no alpha, the alpha value should be dropped.

If converting from a pixel format with no alpha to a pixel format with alpha, the alpha value in the destination pixel format should be set to 1.0.

15.3.15 Luma Keying

Coefficient values to compute the Luma value of the pixel from the pixel components can be specified in the LumaAlphaStruct, along with the upper and lower luma key values. Any pixel that has a luma that falls within the luma key range will have its alpha value zeroed out so that it can be processed appropriately in the blending stage.

Since the pixel values in the pipeline are of a higher precision than the LumaKeyLower and LumaKeyUpper values specified in the Config structure, care should be taken when comparing the pixel luma values against these limits. The LumaKeyLower value should be bit-extended by adding zeroes to the LSBs to match the pixel luma bit width. LumaKeyUpper should be extended by adding ones to the LSBs up to the desired bit width.

15.3.16 Blender

The blender API allows for symmetric blend modes between the VIC and Display units.

The VIC blender supports the following different blending modes:

- DXVAHD_ALPHA_FILL_MODE_OPAQUE
- DXVAHD_ALPHA_FILL_MODE_BACKGROUND
- DXVAHD_ALPHA_FILL_MODE_DESTINATION
- DXVAHD_ALPHA_FILL_MODE_SOURCE_STREAM
- DXVAHD_ALPHA_FILL_MODE_COMPOSITED
- DXVAHD_ALPHA_FILL_MODE_SOURCE_ALPHA

All modes other than _ALPHA_FILL_MODE_COMPOSITED are as defined in the Microsoft DXVA spec and are carried over from VIC 1.0. The new mode that allows the VIC and Display blenders to match each other is enabled by setting the AlphaFillMode in the OutputConfig structureto _ALPHA_FILL_MODE_COMPOSITED. When the AlphaFillMode is set to anything other than _ALPHA_FILL_MODE_COMPOSITED, the srcFact and dstFact parameters specified below are ignored and do not affect the processing.

The parameterizable blender interface allows a variety of blend modes, including the following:

- Per-Pixel Non-Premultiplied Alpha Blend

  output = src*src_alpha + dst*(1-src_alpha)
- Per Pixel Source Premultiplied Alpha Blend

  output = src + dst*(1-src_alpha)
- Constant-Alpha Blend

  output = src*const_alpha + dst*(1-const_alpha)
- Per-Pixel Non-Premultiplied Alpha Blend with constant blend

  output = src*src_alpha*const_alpha + dst*(1-src_alpha*const_alpha)

- Per-Pixel Premultiplied Alpha Blend with constant blend

  output = src*const_alpha + dst*(1-src_alpha*const_alpha)

Note: *ColorKeySelect should always be set to disabled.*

The blend data path will require the color key comparison match result, and per-slot inputs from the config structure to formulate the blend equation.

srcFactC, srcFactA, dstFactC, and dstFactA multiplicand inputs are determined by the config structure programming and color key comparison results.

Table 54: Per-Slot Blend Configuration

Blend Spec Name	Register Name	Description
K1	AlphaK1	10-bit constant alpha value
K2	AlphaK2	10-bit constant alpha value
srcFactC_Match_Select	srcFactCMatchSelect	3-bit enum which sets srcFactC: K1: AlphaK1 K1_TIMES_DST: AlphaK1*dstA NEG_K1_TIMES_DST: 1.0-(AlphaK1*dstA) K1_TIMES_SRC: AlphaK1*srcA ZERO: 0
dstFactC_Match_Select	dstFactCMatchSelect	3-bit enum which sets dstFactC: K1: AlphaK1 K2: AlphaK2 K1_TIMES_DST: AlphaK1*dstA NEG_K1_TIMES_DST: 1.0-(AlphaK1*dstA) NEG_K1_TIMES_SRC: 1.0-(AlphaK1*srcA) ZERO: 0 ONE: 1.0
srcFactA_Match_Select	srcFactAMatchSelect	3-bit enum which sets srcFactA: K1: AlphaK1 K2: AlphaK2 ZERO: 0 NEG_K1_TIMES_DST: 1.0-(AlphaK1*dstA)
dstFactA_Match_Select	dstFactAMatchSelect	3-bit enum which sets dstFactA: K2: AlphaK2 NEG_K1_TIMES_SRC: 1.0-(AlphaK1*srcA) ZERO: 0 ONE: 1.0
UseK3	UseOverrideR	1-bit Boolean to override srcR with constant values K3R
UseK3	UseOverrideG	1-bit Boolean to override srcG with constant values K3G
UseK3	UseOverrideB	1-bit Boolean to override srcB with constant values K3B
UseK3	UseOverrideA	1-bit Boolean to override srcA with constant values K3A
K3R	OverrideR	10-bit source override R value
K3G	OverrideG	10-bit source override G value
K3B	OverrideB	10-bit source override B value
K3A	OverrideA	10-bit source override A value
MaskR	MaskR	1-bit Boolean to override outputR to equal dstR
MaskG	MaskG	1-bit Boolean to override outputG to equal dstG
MaskB	MaskB	1-bit Boolean to override outputB to equal dstB
MaskA	MaskA	1-bit Boolean to override outputA to equal dstA

Given the source and destination factor settings based on config structure inputs and color match results, the per-pixel blend equations are as follows, where srcR, srcG, srcB, srcA, dstR, dstG, dstB, and dstA represent the incoming R, G, B, A components of the incoming source and already present destination pixels in the SFMem buffer (blended results of background and slot[0] through to slot[n-1] for input slot[n]).

```
outputR = (srcFactC * srcR) + (dstFactC * dstR)
outputG = (srcFactC * srcG) + (dstFactC * dstG)
outputB = (srcFactC * srcB) + (dstFactC * dstB)
outputA = (srcFactA * srcA) + (dstFactA * dstA)
```

If the ColorKeySelect is set to DISABLED, then srcFactC, srcFactA, dstFactC, and dstFactA will be programmed by srcFactCMatchSelect, srcFactCMatchSelect, srcFactCMatchSelect, and srcFactCMatchSelect, respectively.

If the ColorKeySelect is set to ENABLED, then srcFactC, srcFactA, dstFactC, and dstFactA are programmed by srcFactCMatchSelect, srcFactCMatchSelect, srcFactCMatchSelect, and srcFactCMatchSelect, respectively, if the destination color key match result returns true, and programmed by srcFactCNoMatchSelect, srcFactCNoMatchSelect, srcFactCNoMatchSelect, and srcFactCNoMatchSelect, respectively, if the destination color key match result returns false.

UseK3 allows the incoming source components to be overridden by constant color components K3R, K3G, K3B, K3A.

MaskR, MaskG, MaskB, and MaskA Boolean settings provide additional per-component mask bits to allow pass-through of the destination pixel components to the output pixel

A programmable background color can still be set to fill the target rectangle background via the OutputConfig structure. The programmed background color should follow these rules:

- The background color is programmed in the same color space as the blending color space. For example, if the blending is happening in an RGB color space, then the background color is treated as an ARGB value. If the blending happens in a YUV color space, the background color is treated as an AYUV value
- Similarly, if the blending happens in a linear color space, the background color is treated as a linear color space value. If the blending is in non-linear space, then the background color is also expected to be non-linear.

15.3.17 Subpixel Source Rectangles

Source rectangles can be specified to a sub-pixel resolution. This feature can be used to smoothly zoom into a portion of an image in real-time, for example, when using the zoom button on a camera. Specifying source and destination rectangles to a pixel resolution quantizes the allowed zoom factor, and hence can cause the zoom to feel jerky. Allowing the source rectangles to be specified to a sub-pixel resolution avoids this problem and makes the zoom smooth.

The config structure is therefore modified to make the source rectangle coordinates U14.16 numbers. Once we have the (decimal) source rectangles, we can calculate adjust the starting phase of the scaling filters accordingly so as to achieve the sub-pixel shift/scale required.

Note that all source coordinates (left, top, right, and bottom) include the bounding pixels. For instance: (0,0)-(0,0) contains 1 pixel, while (0,0)-(1-1) contains a square of 2x2 pixels. This means that rectangles which are less than 1 pixel wide or tall cannot be used. Also, specifying a rectangle of (0.0,0.0)-(0.0,0.1) results in a source rectangle that is 1 pixel wide and 1.1 pixel high, which is not immediately apparent.

15.3.18 Maximum Surface and Rectangle Size Support

The VIC supports surface and rectangle sizes up to 16384x16384 pixels.

This size affects all input and output surfaces parameters as well as all fetch calculations related to these parameters:

- Per-input slot and output surface width/height, luma width/height, chroma width/height
- Per-input slot source and destination rectangle dimensions
- Clear rectangle dimensions
- Output target rectangle dimensions

15.4 Programming Guidelines

The compositor mainly implements the DirectX Video Acceleration 2.0 Enhanced Video Processor specification. This does de-interlacing, scaling, color conversion, proc-amp and compositing for up to 8 input surfaces. It also supports features like gamma correction and non-linear color enhancement.

15.4.1 Class Overview

The software programming interface for the VIC is similar to that of the other video engines in Tegra like NVENC and NVDEC. This compatibility is maintained by virtue of the shared front-ends of both these engines on the Host1x interface, consisting of a THI (Tegra Host Interface) module and a programmable Falcon microcontroller.

15.4.1.1 Method Interface

VIC will have its own class definition, which includes a couple of registers named METHOD_OFFSET and METHOD_DATA, which are interpreted by the THI and are used to create method writes (corresponding to methods in the VIC class specification) into the VIC. These methods will be used to set up pointers to the various surfaces that the VIC needs to read or write, as well as pointers to config structures needed. The VIC class also defines trigger methods such as Execute(), used to start the actual processing of the data for the next output image.

15.4.1.2 Sync-Points

Interaction between hardware and software are controlled and synchronized through the Tegra sync-point mechanism. For example software can request to be notified when the VIC engine is done processing a certain frame's worth of data by enqueuing a sync-point method after the Execute() method that will be sent down for that frame and by programming Host1x to raise an interrupt to the CPU when the sync-point is updated by the VIC hardware.

The THI block is responsible for keeping track of outstanding sync-points and sending appropriate sync-point ACKs back to Host1x. It does this by handling the interrupts raised by the VIC after completing an Execute_Notify() method, and translating that to a sync-point counter update for Host1x.

15.4.1.3 Context-Switch

When programming VIC on GPU products, a hardware interrupt based context switching is used. The Falcon host interface has a context switch state machine that examines incoming context_ptr/valid_flg and performs the correct context switch operation.

Context switching on Tegra devices uses a different mechanism than on GPU devices. We have introduced a new method-based context switching interface to replace the interrupt driven context switching mechanism. To the existing method interface, we add two methods: a context save, and a context restore method. To support the new interface, two changes are required:

- Falcon OS microcode should be modified to act accordingly when receiving either context save or context restore method; it will flag an interrupt when a valid context switch save operation is completed;
- THI interrupt handler should be modified to increment the CTX_SAVE_DONE sync point to Host1x if a valid context save done interrupt is received. This is required to stall Host1x before sending a subsequent context restore method in case any intermediate methods need to be submitted (e.g., ASID switch before restoring the next context)

It is possible that in future Tegra products, multiple ASIDs could exist, and each context could be run with a separate ASID. A Host1x ASID switch method could be pushed after saving the current context, and before restoring the next context.

When using either interrupt-driven context switching, or method-based context switching, VIC supports only WFI (wait-for-idle) context-switching.

The same context switch mechanism can be used to save and restore contexts around power-gating events. When software wishes to power gate the VIC, a context save method should be sent down to save the current context before power gating. On resuming from ELPG, software has to issue a soft-reset to the VIC engine, and then go through the entire re-initialization sequence for VIC, including boot-strapping of the Falcon microcontroller. After the power-up of the engine is complete, software should send a restore method to VIC, which will then cause the saved context to be restored.

Note that VIC is primarily a state-less engine. When processing each frame, methods are issued to the VIC to point to the various surfaces used during each frame's processing, and programming the VIC engine involves writing a configuration structure that specifies all functionality required within the current frame. The engine will read in this configuration structure at the start of every frame of processing. The only state that may require context switch saving would be any pending methods required for upcoming frame executes that have not yet processed by the Falcon microcontroller. There are no security-specific programmable configuration bits in the engine, and thus no concern with handling context switching of security configuration bits.

15.4.1.4 Interrupts

The THI block intercepts interrupts generated by the Falcon to acknowledge the end of the Execute_Notify() methods and translates them to sync-point acknowledges that it then sends out to Host1x. Thus, the only events that cause an interrupt to be generated to the CPU should be fatal events that need software intervention (often a full engine soft-reset) in order to fix.

In VIC 1.0, the Falcon microprocessor launches the VIC hardware on its task and then polls on an idle status register in order to determine when the task is complete. This is inefficient in terms of power and we therefore have added a hardware interrupt which can notify the Falcon microprocessor when all work has been completed. The microprocessor can therefore launch the work and go to a low power wait state, until it is woken up by the interrupt, after which it can service the interrupt and then notify the driver through the sync-point mechanism, etc.

The idle interrupt is enabled by the bit in register NV_CVIC_MISC_INTR_EN_IDLE and will fire when all sub-units of the VIC are idle, and will be hooked up to bit 3 of the Falcon EXT interrupt bus - i.e., the Falcon microcode can mask or check the status of this interrupt using EXT_EXTIRQ4. The programming sequence for this interrupt should be:

1. Falcon microcode does all the VIC initialization and setup. This includes enabling the EXT_EXTIRQ4 interrupt in IRQMASK, setting IRQDEST to route it to the Falcon, and making it an edge-triggered interrupt.
2. Falcon microcode enables the interrupt just before writing the FC_PREPROCESS or FC_COMPOSE registers by writing into NV_CVIC_MISC_INTR_EN_IDLE.
3. Falcon microcode then enters a wait state.
4. On receiving the interrupt, Falcon microcode wakes up, clears the interrupt by writing into NV_CVIC_FALCON_IRQSCLR_EXT_EXTIRQ4 and then disables the interrupt by writing 0x0 into NV_CVIC_MISC_INTR_EN_IDLE.
5. The rest of the Falcon microcode sequence completes as before, including generating any EXECUTE_AWAKEN interrupts to host, etc.

15.4.1.5 Driver Programming Model

The compositor operation requires the following buffers to be provided. These buffers will be explained in more detail in the following sections. When the Execute() method is sent, the Falcon program will be initiated which will in turn trigger the various VIC operations in the correct sequence.

Buffers	I/O	Producer	Description/Restrictions
Config Buffer	I	Driver	Configuration parameters
Palette Buffer	I	Driver	Palette table
History Buffer	I/O	App	Histogram and Cadence parameters
Input surfaces	I	Driver/VIC	Input Surfaces
Noise reduced surfaces	I/O	VIC	Noise reduced version of input
Motion map surfaces	I/O	VIC	Motion map of input
Output surface	O	VIC	Final output surface (post-composition)
FCE Ucode buffer	I	Driver	FCE microcode
CrcStruct Offset	O	VIC	CRC struct

15.4.1.6 Surface Padding and Alignment

The base address of all buffers (input and output) programmed into the VIC have to be aligned to a multiple of the block size for block-linear surfaces, and a multiple of 256B for pitch linear surfaces. Thus, the buffer base addresses will always be aligned to at least 256B; the base address pointers programmed into the VIC reflect this and only accept address bits [39:8].

The parameters SurfaceWidth and SurfaceHeight used in the ConfigStruct refer to the actual image dimensions. LumaWidth and LumaHeight refer to the padded buffer dimensions, and apply to both luma and RGBA surfaces. ChromaWidth and ChromaHeight refer to the padded buffer dimensions for the chroma planes of planar or semi-planar YUV surfaces. All the Surface/Luma/Chroma widths and heights are programmed in terms of pixels, not bytes. The constraints below describe the restrictions on the buffer widths and heights.

The VIC does not assume that the widths of block-linear buffers are automatically padded to the next multiple of 64B. The VIC Luma/Chroma Width fields are still used to describe the padded surface pitch for the block-linear buffer, and can be any multiple of the block width.

When using VIC configurable surface cache entry modes of 16Bx16, 32Bx8 or 64Bx4, VIC can read in any block linear surface. Note, however, the VIC cache entry mode of 128Bx2 will only work where the buffer widths are multiples of 128B.

The constraint for any image buffer size is for it to be a multiple of the surface memory format block size as well as the configurable surface cache entry size. The memory format block size is considered to be 256x1 for pitch linear, and 64x(8<<BLK_HEIGTH) for block linear buffers. BLK_HEIGHT is the log2 encoded block height as usual.

For input surfaces:

- If pitch linear, then the configurable surface cache entry size can be set to 256Bx1, 128Bx2, or 64Bx4. The buffer width should be padded to a multiple of 256B, and the height should be padded to a multiple of the configurable surface cache entry height.
- If block linear, then the configurable surface cache entry size can be set to 128Bx2, 64Bx4, 32Bx8 or 16Bx16. The buffer width should be padded to the next multiple of the max of 64B or the configurable surface cache entry width, and the height should be padded to the max of the configurable surface cache entry height and the block height in lines (8 << BLK_HEIGHT, where BLK_HEIGHT is the number of GOBs tall per block linear block, log2 encoded).

For output surfaces:

- If pitch linear, the buffer width should be padded to a multiple of 256B.
- If block linear, the buffer width should be padded to the next multiple of 64B, and the height should be padded to a multiple of the block height in lines (8 << BLK_HEIGHT, where BLK_HEIGHT is the number of GOBs tall per block linear block, log2 encoded).

As noted previously, the VIC has to process 16Bx16 blocks of memory in the pre-process phase. This would be very inefficient to do if the surfaces are set up as pitch-linear buffers, and therefore the usage of pitch-linear buffers for images that need pre-processing is strongly discouraged. For block linear buffers, the Falcon microcode forces the cache-line dimensions for the pre-process phase alone to 16Bx16 to increase efficiency. As a result of this, software should make sure to make the image buffer height a multiple of 16 when enabling pre-processing.

While any pixel data outside of the boundaries defined by SurfaceWidth/SurfaceHeight of an input surface will not be used to contribute to the output image, it may still be read into the VIC surface cache. If the image buffer is padded as described above, the hardware will not issue any read or write accesses outside of the image buffer range.

15.4.1.7 Bootstrapping the VIC Falcon

The process of initializing and bootstrapping involves loading a portion of the Falcon microcode using register triggered hardware DMAs and then starting the Falcon microcontroller to execute this microcode. This bootstrap code in turn fetches all the remaining required microcode from memory and continues execution. This procedure will have to be followed both on boot and while resuming from a power-gating event.

On boot, if the VIC must be used for splash screen rendering, the Falcon should be bootstrapped by the boot loader. The VIC Falcon microcode binary may be embedded into the boot loader binary.

Once in the OS, the Falcon may be re-bootstrapped by a driver, with a more recent version of the Falcon microcode. Under Windows, bootstrapping will likely be the responsibility of the GPU Resource Manager (RM). Under Linux/Android, bootstrapping will likely be the responsibility of the Linux host driver.

OS Overview

The MSDEC OS is a lightweight OS that runs on Falcon. It is independent of the engine (VLD, PDEC, PPP, or SEC). The OS takes care of the following functionalities:

- Loading itself from boot-loader code.
- Setting up the IRQDEST and IRQMASK registers to demarcate which interrupts are handled by host and which by Falcon.
- Setting up Interrupt handlers 0 and 1 and exception handler.
- Processing method and context switch interrupts
- One receiving an execute method, loading the correct application code for the engine, and triggering execution of application code.
- Notifying host of any application error (Mailbox0 specifies that it is an application error, and Mailbox1 contains error code).
- Notifying host of any other errors, as specified in the msdecos_host_interface.h header file.

OS State Machine

The OS during a typical sequence of operation goes through the following states:

1. RM loads 256 bytes of OS IMEM and DMEM through PRIV writes.
2. OS loads the rest of OS code through imreads.
3. Initializes the engine.
4. Handles any incoming context switch and method interrupts.
5. On receiving an execute method, calls app and hand over execution to app.
6. Once app is done, sets engine to wait mode and wait for interrupts.

If there are any error/exception interrupts during the course of execution of the app (or OS) then the interrupt handlers raise interrupt to host with an appropriate error code and wait for the interrupt to be cleared before proceeding.

OS boot up and initialization sequence

The OS resident code fits into 768 bytes and the OS resident data into 512 bytes. Of the resident data, the first 256 contain global constants that might be initialized to non-zero values, but the bytes 256-511 are all initialized to zero at the time of context switch and need not be loaded in explicitly through dmreads.

RM loads the first 256 bytes of IMEM and DMEM through PRIV writes. The first 256 bytes of OS code loads the remaining 512 bytes through IMEM reads.

Initialization accomplishes the following:

1. OS writes its version number to DWORD 0 in DMEM, which can be used for debug purposes.
2. The watchdog timer is disabled by setting bit zero of WDTMRCTL to zero.
3. Interrupt vectors and exception vector are initialized.
4. IRQMASK and IRQDEST initialized to appropriate values. See Chapter 3: Interrupt Controller for details.
5. Enable interrupts 0 and 1 to start process interrupts

15.4.2 Application Enumerations

Below are the enumerators used in the configuration structures passed by the driver:

- DXVAHD_FRAME_FORMAT
- PIXEL_FORMAT
- DXVAHD_DEINTERLACE_MODE_PRIVATE
- DXVAHD_ALPHA_FILL_MODE
- BLK_KIND
- BLEND_SRCFACTC
- BLEND_DSTFACTC
- BLEND_SRCFACTA
- BLEND_DSTFACTA
- GAMMA_MODE
- FILTER_LENGTH
- FILTER_TYPE

Table 55: DXVAHD_FRAME_FORMAT

Enumerant	Value	Description
DXVAHD_FRAME_FORMAT_PROGRESSIVE	0	A_PROGRESSIVE frame format refers to a non-interlaced surface. The only deinterlacing mode that is valid for this frame format is _WEAVE.
DXVAHD_FRAME_FORMAT_INTERLACED_TOP_FIELD_FIRST	1	An interlaced surface, but the top and bottom fields (from different time instants) are weaved together into a single input surface with the top field on the first and every alternating line. The valid deinterlacing modes for this frame format are _WEAVE and _BOB.
DXVAHD_FRAME_FORMAT_INTERLACED_BOTTOM_FIELD_FIRST	2	An interlaced surface, but the top and bottom fields (from different time instants) are weaved together into a single input surface with the bottom field on the first line. The valid deinterlacing modes for this frame format are _WEAVE and _BOB.
DXVAHD_FRAME_FORMAT_TOP_FIELD	3	A surface with frame format _TOP_FIELD is a field surface and is hence half the height of the original progressive source. The lines in the current surface should be displayed above the lines of the previous field (which would be of type _BOTTOM_FIELD). The valid deinterlacing modes for this frame format are _WEAVE, _BOB_FIELD, _NEWBOB, _DISI1 and _WEAVE_LUMA_BOB_FIELD_CHROMA.
DXVAHD_FRAME_FORMAT_BOTTOM_FIELD	4	A surface with frame format _BOTTOM_FIELD is a field surface and is hence half the height of the original progressive source. The lines in the current surface should be displayed below the lines of the previous field (which would be of type _TOP_FIELD). The valid deinterlacing modes for this frame format are _WEAVE, _BOB_FIELD, _NEWBOB, _DISI1 and _WEAVE_LUMA_BOB_FIELD_CHROMA.
DXVAHD_FRAME_FORMAT_SUBPIC_PROGRESSIVE	5	The _SUBPIC frame formats are similar to the non-subpic frame formats, but are meant for the sub-picture layers from the blu-ray spec. Choosing a sub-picture vs. a non-sub-picture frame format affects the filter kernels that are used to scale the slot.
DXVAHD_FRAME_FORMAT_SUBPIC_INTERLACED_TOP_FIELD_FIRST	6	

Table 55: DXVAHD_FRAME_FORMAT

Enumerant	Value	Description
DXVAHD_FRAME_FORMAT_SUBPIC_INTERLACED_BOTTOM_FIELD_FIRST	7	
DXVAHD_FRAME_FORMAT_SUBPIC_TOP_FIELD	8	
DXVAHD_FRAME_FORMAT_SUBPIC_BOTTOM_FIELD	9	
DXVAHD_FRAME_FORMAT_TOP_FIELD_CHROMA_BOTTOM	10	A field surface with the luma corresponding to the _TOP_FIELD, but with chroma corresponding to the _BOTTOM_FIELD. The valid deinterlacing modes for this frame format are _WEAVE, _BOB_FIELD, _NEWBOB, _DISI1 and _WEAVE_LUMA_BOB_FIELD_CHROMA.
DXVAHD_FRAME_FORMAT_BOTTOM_FIELD_CHROMA_TOP	11	A field surface with the luma corresponding to the _BOTTOM_FIELD, but with chroma corresponding to the _TOP_FIELD. The valid deinterlacing modes for this frame format are _WEAVE, _BOB_FIELD, _NEWBOB, _DISI1 and _WEAVE_LUMA_BOB_FIELD_CHROMA.
DXVAHD_FRAME_FORMAT_SUBPIC_TOP_FIELD_CHROMA_BOTTOM	12	
DXVAHD_FRAME_FORMAT_SUBPIC_BOTTOM_FIELD_CHROMA_TOP	13	

Table 56: PIXEL_FORMAT

Enumerant	Value
T_A8	0
T_L8	1
T_A4L4	2
T_L4A4	3
T_R8	4
T_A8L8	5
T_L8A8	6
T_R8G8	7
T_G8R8	8
T_B5G6R5	9
T_R5G6B5	10
T_B6G5R5	11
T_R5G5B6	12
T_A1B5G5R5	13
T_A1R5G5B5	14
T_B5G5R5A1	15
T_R5G5B5A1	16
T_A5B5G5R1	17
T_A5R1G5B5	18
T_B5G5R1A5	19
T_R1G5B5A5	20
T_X1B5G5R5	21
T_X1R5G5B5	22
T_B5G5R5X1	23
T_R5G5B5X1	24
T_A4B4G4R4	25
T_A4R4G4B4	26
T_B4G4R4A4	27

Table 56: PIXEL_FORMAT

Enumerant	Value
T_R4G4B4A4	28
T_B8_G8_R8	29
T_R8_G8_B8	30
T_A8B8G8R8	31
T_A8R8G8B8	32
T_B8G8R8A8	33
T_R8G8B8A8	34
T_X8B8G8R8	35
T_X8R8G8B8	36
T_B8G8R8X8	37
T_R8G8B8X8	38
T_A2B10G10R10	39
T_A2R10G10B10	40
T_B10G10R10A2	41
T_R10G10B10A2	42
T_A4P4	43
T_P4A4	44
T_P8A8	45
T_A8P8	46
T_P8	47
T_P1	48
T_U8V8	49
T_V8U8	50
T_A8Y8U8V8	51
T_V8U8Y8A8	52
T_Y8_U8_V8	53
T_Y8_V8_U8	54
T_U8_V8_Y8	55
T_V8_U8_Y8	56
T_Y8_U8__Y8_V8	57
T_Y8_V8__Y8_U8	58
T_U8_Y8__V8_Y8	59
T_V8_Y8__U8_Y8	60
T_Y8___U8V8_N444	61
T_Y8___V8U8_N444	62
T_Y8___U8V8_N422	63
T_Y8___V8U8_N422	64
T_Y8___U8V8_N422R	65
T_Y8___V8U8_N422R	66
T_Y8___U8V8_N420	67
T_Y8___V8U8_N420	68
T_Y8___U8___V8_N444	69
T_Y8___U8___V8_N422	70
T_Y8___U8___V8_N422R	71
T_Y8___U8___V8_N420	72
T_U8	73

Table 56: PIXEL_FORMAT

Enumerant	Value
T_V8	74

Table 57: DXVAHD_DEINTERLACE_MODE_PRIVATE

Enumerant	Value	Description
DXVAHD_DEINTERLACE_MODE_PRIVATE_WEAVE	0	Weave the top and bottom fields together. Requires previous and current field surfaces to be enabled.
DXVAHD_DEINTERLACE_MODE_PRIVATE_BOB_FIELD	1	Simply double the current field in height. Only requires current field to be enabled.
DXVAHD_DEINTERLACE_MODE_PRIVATE_BOB	2	Simply double the current field in height. Only works with the _INTERLACED frame formats, and picks the current field lines out of the input interleaved surface. Only requires current field to be enabled.
DXVAHD_DEINTERLACE_MODE_PRIVATE_NEWBOB	3	Similar to the BOB_FIELD mode, but uses a motion adaptive algorithm to decide whether to BOB or WEAVE fields together. Requires at least the following fields enabled: previous, next, current, previous motion, motion, combined motion.
DXVAHD_DEINTERLACE_MODE_PRIVATE_DISI1	4	Uses a motion adaptive algorithm to decide whether to BOB or WEAVE fields together. In addition, uses an edge-directed spatial interpolation algorithm in order to avoid any weave artifacts. Requires at least the following fields enabled: previous, next, current, previous motion, motion, combined motion.
DXVAHD_DEINTERLACE_MODE_PRIVATE_WEAVE_LUMA_BOB_FIELD_CHROMA	5	Uses the WEAVE algorithm on the luma, and the BOB_FIELD algorithm on the chroma components.
DXVAHD_DEINTERLACE_MODE_PRIVATE_MAX	15	This is not a valid mode for deinterlacing and should not be programmed into the hardware.

Table 58: DXVAHD_ALPHA_FILL_MODE

Enumerant	Value	Description
DXVAHD_ALPHA_FILL_MODE_OPAQUE	0	Opaque (all alpha inside target rect will be set to 1.0).
DXVAHD_ALPHA_FILL_MODE_BACKGROUND	1	Background (all alpha inside target rect will be set to background alpha).
DXVAHD_ALPHA_FILL_MODE_DESTINATION	2	Destination (alpha will remain unchanged).
DXVAHD_ALPHA_FILL_MODE_SOURCE_STREAM	3	Source stream (alpha from source stream without planar alpha).
DXVAHD_ALPHA_FILL_MODE_COMPOSITED	4	Composited (composited alpha, starting with background). In this mode, the blend parameters specified by SrcFactA/DstFactA/ SrcFactC/DstFactC etc. are used.
DXVAHD_ALPHA_FILL_MODE_SOURCE_ALPHA	5	Source alpha (alpha from source stream with planar alpha).

Table 59: BLK_KIND

Enumerant	Value	Description
BLK_KIND_PITCH	0	This enum is used to program the various "BlkKind" fields in the ConfigSructure.
BLK_KIND_GENERIC_16Bx2	1	Tegra Block Linear (16Bx2) memory format

Table 59: BLK_KIND

Enumerant	Value	Description
BLK_KIND_BL_NAIVE	2	This format is not supported in the VIC hardware and should not be programmed in the config structure.
BLK_KIND_BL_KEPLER_XBAR_RAW	3	This format is not supported in the VIC hardware and should not be programmed in the config structure.
BLK_KIND_VP2_TILED	15	This format is not supported in the VIC hardware and should not be programmed in the config structure.

Table 60: BLEND_SRCFACTC

Enumerant	Value	Description
BLEND_SRCFACTC_K1	0	SrcFactC = AlphaK1
BLEND_SRCFACTC_K1_TIMES_DST	1	SrcFactC = AlphaK1 * dstA
BLEND_SRCFACTC_NEG_K1_TIMES_DST	2	SrcFactC = 1.0 - (AlphaK1 * dstA)
BLEND_SRCFACTC_K1_TIMES_SRC	3	SrcFactC = AlphaK1 * srcA
BLEND_SRCFACTC_ZERO	4	SrcFactC = 0

Table 61: BLEND_DSTFACTC

Enumerant	Value	Description
BLEND_DSTFACTC_K1	0	DstFactC = AlphaK1
BLEND_DSTFACTC_K2	1	DstFactC = AlphaK2
BLEND_DSTFACTC_K1_TIMES_DST	2	DstFactC = AlphaK1 * dstA
BLEND_DSTFACTC_NEG_K1_TIMES_DST	3	DstFactC = 1.0 - (AlphaK1 * dstA)
BLEND_DSTFACTC_NEG_K1_TIMES_SRC	4	DstFactC = 1.0 - (AlphaK1 * srcA)
BLEND_DSTFACTC_ZERO	5	DstFactC = 0.0
BLEND_DSTFACTC_ONE	6	DstFactC = 1.0

Table 62: BLEND_SSTFACTA

Enumerant	Value	Description
BLEND_SRCFACTA_K1	0	SrcFactA = AlphaK1
BLEND_SRCFACTA_K2	1	SrcFactA = AlphaK2
BLEND_SRCFACTA_NEG_K1_TIMES_DST	2	SrcFactA = 1.0 - (AlphaK1 * dstA)
BLEND_SRCFACTA_ZERO	3	SrcFactA = 0.0
BLEND_SRCFACTA_MAX	7	Invalid mode - should not be programmed.

Table 63: BLEND_DSTFACTA

Enumerant	Value	Description
BLEND_DSTFACTA_K2	0	DstFactA = AlphaK2
BLEND_DSTFACTA_NEG_K1_TIMES_SRC	1	DstFactA = 1.0 - (AlphaK1 * srcA)
BLEND_DSTFACTA_ZERO	2	DstFactA = 0.0
BLEND_DSTFACTA_ONE	3	DstFactA = 1.0
BLEND_DSTFACTA_MAX	7	Invalid mode - should not be programmed.

Table 64: GAMMA_MODE

Enumerant	Value	Description
GAMMA_MODE_NONE	0	Do not apply any degamma or regamma to the color components.
GAMMA_MODE_SRGB	1	Apply the sRGB degamma/regamma curve to the color components.

Table 64: GAMMA_MODE

Enumerant	Value	Description
GAMMA_MODE_REC709	2	Apply the Rec.709 degamma/regamma curve to the color components.
GAMMA_MODE_RESERVED	3	Apply the Rec.2020 degamma/regamma curve to the color components.

Table 65: FILTER_LENGTH

Enumerant	Value	Description
FILTER_LENGTH_1TAP	0	Nearest-neighbour filtering
FILTER_LENGTH_2TAP	1	Bi-linear filtering
FILTER_LENGTH_5TAP	2	5-tap filters with filter kernel specified using the FilterStruct
FILTER_LENGTH_10TAP	3	10-tap filters with filter kernel specified using the FilterStruct

Table 66: FILTER_TYPE

Enumerant	Value	Description
FILTER_TYPE_NORMAL	0	Filter kernel that is used for regular scaling.
FILTER_TYPE_NOISE	1	Filter kernel that is used for spatial smoothing. The weight of this filter is specified using the FilterNoise/ChromaNoise parameters in the SlotConfig structure.
FILTER_TYPE_DETAIL	2	Filter kernel that is used for spatial sharpening. The weight of this filter is specified using the FilterDetail/ChromaDetail parameters in the SlotConfig structure.

15.4.3 Application Memory Structures

15.4.3.1 Config Structure

The Config structure is read from the frame buffer by the surface cache unit and is stored in small memories inside each subunit. The entire ConfigStruct is the concatenation of the Surface Cache, Surface List, Color Conversion, Blending and Fetch Control structs (in this order). The structs are read in 128-bit units from memory and sent to the according subunits memory with the correct memory address by the Surface Cache.

The config structure is broken up into smaller structs that are used depending on the active slots and their content. Each structure is a multiple of 128 bits in size, and the start of the struct needs to be aligned to a 256 byte boundary.

Note: NV12 and NV24 share the same pixel format but can be distinguished by the fact that NV24 is a field based format (i.e., every surface contains a single field) whereas NV12 is frame based (i.e., every surface contains an entire frame).

Note: For highest quality, filter override mode as defined in FetchControlCoeffStruct should always be used.

The following table defines the smaller structs that comprise ConfigStruct.

Table 67: ConfigStruct Substructures

Name	Data Type	Offset	Notes
pipeConfig	PipeConfig (128 bits)	0	
outputConfig	OutputConfig (128 bits)	128	
outputSurfaceConfig	OutputSurfaceConfig (128 bits)	256	
outColorMatrixStruct	MatrixStruct (256 bits)	384	
clearRectStruct[4]	ClearRectStruct (128 bits)	640 + (i * 128)	Up to eight clear rectangles supported. Each ClearRectStruct specifies two rectangles
slotStruct[8]	SlotStruct (1408 bits)	1152 + (i * 1408)	Per-slot configuration structures

pipeConfig

Table 68: PipeConfig Details

Name	Data Type	Offset	Notes
DownsampleHoriz	fixed<0,11,0> (11 bits)	0	TargetWidth/DownsampleWidth (U9.2 used to load bias filter) Set to 0 to enable filter override mode (mapping between stream and filter explained below)
reserved0	fixed<0,5,0> (5 bits)	11	
DownsampleVert	fixed<0,11,0> (11 bits)	16	TargetHeight/DownsampleHeight (U9.2 used to load bias filter) Set to 0 to enable filter override mode (mapping between stream and filter explained below)
reserved1	fixed<0,5,0> (5 bits)	27	
reserved2	fixed<0,32,0> (32 bits)	32	
reserved3	fixed<0,32,0> (32 bits)	64	
reserved4	fixed<0,32,0> (32 bits)	96	

outputConfig

This structure specifies the output flip enables and Target rectangle.

Table 69: outputConfig Details

Name	Data Type	Offset	Notes
AlphaFillMode	fixed<0,3,0> (3 bits)	0	Alpha fill mode (DXVAHD_ALPHA_FILL_MODE)
AlphaFillSlot	fixed<0,3,0> (3 bits)	3	SlotId for when AlphaFillMode == Source stream/Source alpha
BackgroundAlpha	fixed<0,10,0> (10 bits)	6	Background color A
BackgroundR	fixed<0,10,0> (10 bits)	16	Background color R
BackgroundG	fixed<0,10,0> (10 bits)	26	Background color G
BackgroundB	fixed<0,10,0> (10 bits)	36	Background color B
RegammaMode	fixed<0,2,0> (2 bits)	46	The regamma curve to be used for the blended output
OutputFlipX	fixed<0,1,0> (1 bit)	48	Horizontal flip enable
OutputFlipY	fixed<0,1,0> (1 bit)	49	Vertical flip enable
OutputTranspose	fixed<0,1,0> (1 bit)	50	Transpose enable
reserved1	fixed<0,1,0> (1 bit)	51	
reserved2	fixed<0,12,0> (12 bits)	52	
TargetRectLeft	fixed<0,14,0> (14 bits)	64	Target rectangle. Restricts the output to a certain rectangle inside the output surface. Pixels outside of this area are guaranteed to remain unmodified.

Table 69: outputConfig Details

Name	Data Type	Offset	Notes
reserved3	fixed<0,2,0> (2 bits)	78	
TargetRectRight	fixed<0,14,0> (14 bits)	80	
reserved4	fixed<0,2,0> (2 bits)	94	
TargetRectTop	fixed<0,14,0> (14 bits)	96	
reserved5	fixed<0,2,0> (2 bits)	110	
TargetRectBottom	fixed<0,14,0> (14 bits)	112	
reserved6	fixed<0,2,0> (2 bits)	126	

outputSurfaceConfig

Table 70: outputSurfaceConfig

Name	Data Type	Offset	Notes
OutPixelFormat	fixed<0,7,0> (7 bits)	0	Pixel format of output surface (PIXEL_FORMAT)
OutChromaLocHoriz	fixed<0,2,0> (2 bits)	7	Chroma location of output surface (See SurfaceListSurfaceStruct)
OutChromaLocVert	fixed<0,2,0> (2 bits)	9	
OutBlkKind	fixed<0,4,0> (4 bits)	11	The block linear kind of the output surface (BLK_KIND)
OutBlkHeight	fixed<0,4,0> (4 bits)	15	Block-linear height (in gobs, log2)
reserved0	fixed<0,3,0> (3 bits)	19	
reserved1	fixed<0,10,0> (10 bits)	22	
OutSurfaceWidth	fixed<0,14,0> (14 bits)	32	Output surface width minus 1
OutSurfaceHeight	fixed<0,14,0> (14 bits)	46	Output surface height minus 1
reserved2	fixed<0,4,0> (4 bits)	60	
OutLumaWidth	fixed<0,14,0> (14 bits)	64	Padded output surface luma width minus 1
OutLumaHeight	fixed<0,14,0> (14 bits)	78	Padded output surface luma height minus 1
reserved3	fixed<0,4,0> (4 bits)	92	
OutChromaWidth	fixed<0,14,0> (14 bits)	96	Padded output surface chroma width minus 1
OutChromaHeight	fixed<0,14,0> (14 bits)	110	Padded output surface chroma height minus 1
reserved4	fixed<0,4,0> (4 bits)	124	

outColorMatrixStruct

The matrices defined in these structures are used to specify the color space conversion between input and output pixel formats if any.

Table 71: outColorMatrixStruct Details

Name	Data Type	Offset	Notes
matrix_coeff00	fixed<0,20,0> (20 bits)	0	Matrix entry (0,0) of 4x3 color conversion matrix. Precision and right shift are the same as for the luma vector.
matrix_coeff10	fixed<0,20,0> (20 bits)	20	Matrix entry (1,0) of 4x3 color conversion matrix
matrix_coeff20	fixed<0,20,0> (20 bits)	40	Matrix entry (2,0) of 4x3 color conversion matrix
matrix_r_shift	fixed<0,4,0> (4 bits)	60	Right shift value for matrix
matrix_coeff01	fixed<0,20,0> (20 bits)	64	Matrix entry (0,1) of 4x3 color conversion matrix
matrix_coeff11	fixed<0,20,0> (20 bits)	84	Matrix entry (1,1) of 4x3 color conversion matrix
matrix_coeff21	fixed<0,20,0> (20 bits)	104	Matrix entry (2,1) of 4x3 color conversion matrix
reserved0	fixed<0,3,0> (3 bits)	124	
matrix_enable	fixed<0,1,0> (1 bit)	127	
matrix_coeff02	fixed<0,20,0> (20 bits)	128	Matrix entry (0,2) of 4x3 color conversion matrix
matrix_coeff12	fixed<0,20,0> (20 bits)	148	Matrix entry (1,2) of 4x3 color conversion matrix
matrix_coeff22	fixed<0,20,0> (20 bits)	168	Matrix entry (2,2) of 4x3 color conversion matrix
reserved1	fixed<0,4,0> (4 bits)	188	
matrix_coeff03	fixed<0,20,0> (20 bits)	192	Matrix entry (0,3) of 4x3 color conversion matrix
matrix_coeff13	fixed<0,20,0> (20 bits)	212	Matrix entry (1,3) of 4x3 color conversion matrix
matrix_coeff23	fixed<0,20,0> (20 bits)	232	Matrix entry (2,3) of 4x3 color conversion matrix
reserved2	fixed<0,4,0> (4 bits)	252	

ClearRectStruct[4]

The ClearRectStruct structures together define 8 clear rectangles. These rectangles are enabled or disabled for each slot using the fields in SurfaceList0Struct.

Table 72: ClearRectStruct[4] Details

Name	Data Type	Offset	Notes
ClearRect0Left	fixed<0,14,0> (14 bits)	0	First clear rectangle of 128-bit chunk
reserved0	fixed<0,2,0> (2 bits)	14	
ClearRect0Right	fixed<0,14,0> (14 bits)	16	

Table 72: ClearRectStruct[4] Details

Name	Data Type	Offset	Notes
reserved1	fixed<0,2,0> (2 bits)	30	
ClearRect0Top	fixed<0,14,0> (14 bits)	32	
reserved2	fixed<0,2,0> (2 bits)	46	
ClearRect0Bottom	fixed<0,14,0> (14 bits)	48	
reserved3	fixed<0,2,0> (2 bits)	62	
ClearRect1Left	fixed<0,14,0> (14 bits)	64	Second clear rectangle of 128-bit chunk
reserved4	fixed<0,2,0> (2 bits)	78	
ClearRect1Right	fixed<0,14,0> (14 bits)	80	
reserved5	fixed<0,2,0> (2 bits)	94	
ClearRect1Top	fixed<0,14,0> (14 bits)	96	
reserved6	fixed<0,2,0> (2 bits)	110	
ClearRect1Bottom	fixed<0,14,0> (14 bits)	112	
reserved7	fixed<0,2,0> (2 bits)	126	

slotStruct[8]

Table 73: slotStruct[8] Details

Name	Data Type	Offset
slotConfig	SlotConfig (512 bits)	0
slotSurfaceConfig	SlotSurfaceConfig (128 bits)	512
lumaKeyStruct	LumaKeyStruct (128 bits)	640
colorMatrixStruct	MatrixStruct (256 bits)	768
gamutMatrixStruct	MatrixStruct (256 bits)	1024
blendingSlotStruct	BlendingSlotStruct (128 bits)	1280

SlotConfig

This structure contains enable bits for each slot, which need to be set as described below.

- Noise reduction requires a forward and backward reference field for interlaced content and a backward reference for progressive content (see Methods section about how to set surfaces). Both require a noise reduced surface (field for interlaced, frame for progressive).
- MotionMap calculation is required for DiSi1/DiNewBob and also needs a forward and backward reference. Behavior is not defined for progressive streams so enabling it needs to be flagged as an error.
- Cadence Detection will enable artifact and weave counts. It will also enable Falcon code to force deinterlace mode to weave if a cadence was detected.

Table 74: SlotConfig Details

Name	Data Type	Offset	Description
SlotEnable	fixed<0,1,0> (1 bit)	0	Enable or disable bit for this slot
DeNoise	fixed<0,1,0> (1 bit)	1	Enable bit for noise reduction
AdvancedDenoise	fixed<0,1,0> (1 bit)	2	Enable the advanced denoising algorithm (TNR2) for each slot.
CadenceDetect	fixed<0,1,0> (1 bit)	3	Cadence detection enable bit
MotionMap	fixed<0,1,0> (1 bit)	4	Motion map enable bit for each slot. Required for DiSi1/ DiNewBob
MMapCombine	fixed<0,1,0> (1 bit)	5	Enable bit for combine motion map for each slot. This is required for DiSi1/DiNewBob. This requires MotionMap being enabled and also required a prevMotionMap surface (see Methods). Behavior without MotionMap enabled is undefined and has to be flagged as an error.
IsEven	fixed<0,1,0> (1 bit)	6	Select if current field is even (used for cadence detection)
ChromaEven	fixed<0,1,0> (1 bit)	7	Enable if chroma of current field is even
CurrentFieldEnable	fixed<0,1,0> (1 bit)	8	Indicates which surfaces are enabled for fetching Bit 0: Current field Bit 1: Previous field Bit 2: Next field Bit 3: Next field (noise filtered, takes priority over unfiltered field) Bit 4: Current motion field Bit 5: Previous motion field Bit 6: Previous previous motion field Bit 7: Combined motion field
PrevFieldEnable	fixed<0,1,0> (1 bit)	9	
NextFieldEnable	fixed<0,1,0> (1 bit)	10	
NextNrFieldEnable	fixed<0,1,0> (1 bit)	11	
CurMotionFieldEnable	fixed<0,1,0> (1 bit)	12	
PrevMotionFieldEnable	fixed<0,1,0> (1 bit)	13	
PpMotionFieldEnable	fixed<0,1,0> (1 bit)	14	
CombMotionFieldEnable	fixed<0,1,0> (1 bit)	15	
FrameFormat	fixed<0,4,0> (4 bits)	16	Frame format of the frame (DXVAHD_FRAME_FORMAT)
FilterLengthY	fixed<0,2,0> (2 bits)	20	Filter length for each stream: 0: 1-tap (nearest neighbor) 1: 2-tap (bi-linear) 2: 5-tap 3: 10-tap
FilterLengthX	fixed<0,2,0> (2 bits)	22	
Panoramic	fixed<0,12,0> (12 bits)	24	Panoramic scaling parameter.

Table 74: SlotConfig Details

Name	Data Type	Offset	Description
reserved1	fixed<0,22,0> (22 bits)	36	
DetailFltClamp	fixed<0,6,0> (6 bits)	58	The maximum range allowed for the difference between the filter output and center pixel of the input filter support. Set to 0 to disable clamping.
FilterNoise	fixed<0,10,0> (10 bits)	64	Strength of the spatial noise filter for slot 0. All detail and noise filter values (including chroma) become meaningless and should be set to 0 as soon as filter override is enabled
FilterDetail	fixed<0,10,0> (10 bits)	74	Strength of the detail filter for slot 0
ChromaNoise	fixed<0,10,0> (10 bits)	84	Strength of the noise filter for slot 0
ChromaDetail	fixed<0,10,0> (10 bits)	94	Strength of the detail filter for slot 0
DeinterlaceMode	fixed<0,4,0> (4 bits)	104	Deinterlace mode (DXVAHD_DEINTERLACE_MODE_PRIVATE)
MotionAccumWeight	fixed<0,3,0> (3 bits)	108	Accumulation weight for motion IIR filter for slot 0 (default should be 6). The first time an even/odd motion field is calculated AccumWeight should be set to 0 to avoid having to clear the motion buffer beforehand.
Noiselir	fixed<0,11,0> (11 bits)	111	Accumulation weight for noise reduction IIR filter for slot 0 (default value should be 0x300).
LightLevel	fixed<0,4,0> (4 bits)	122	Describes the level of lighting present when the input image was captured. This parameter is used along with the AdvancedDenoise bits to determine the exact denoising algorithm to be applied for noise reduction.
reserved4	fixed<0,2,0> (2 bits)	126	
SoftClampLow	fixed<0,10,0> (10 bits)	128	Lower soft clamping bound
SoftClampHigh	fixed<0,10,0> (10 bits)	138	Upper soft clamping bound
reserved5	fixed<0,3,0> (3 bits)	148	
reserved6	fixed<0,9,0> (9 bits)	151	
PlanarAlpha	fixed<0,10,0> (10 bits)	160	10-bit Planar alpha value. This planar alpha will be multiplied with the alpha value coming from the stream. In case of palettized alpha formats (like AI44/A8P8/AI88) the stream alpha value is the alpha value from the surface multiplied with the alpha value from the palette.
ConstantAlpha	fixed<0,1,0> (1 bit)	170	If true, planar alpha value will be used instead of stream alpha, if false, stream alpha will be multiplied with planar alpha. Constant alpha has to be set for all surfaces not containing any alpha data (like XRGB/NV12/YUY2/UYVY/YV12).
StereoInterleave	fixed<0,3,0> (3 bits)	171	Enable pixel interleave for auto-stereoscopic panels (STEREO_INTERLEAVE)
ClipEnabled	fixed<0,1,0> (1 bit)	174	Enable clip against negative values after gamut matrix
ClearRectMask	fixed<0,8,0> (8 bits)	175	Clear rectangle mask
DegammaMode	fixed<0,2,0> (2 bits)	183	The de-gamma curve to be used for the slot (GAMMA_MODE)
reserved7	fixed<0,1,0> (1 bit)	185	

Table 74: SlotConfig Details

Name	Data Type	Offset	Description
DecompressEnable	fixed<0,1,0> (1 bit)	186	Decompression enable bit for the slot. If set to false, then all requests will bypass the CDE and go directly to MCCIF. If this bit is set, then all requests to the surface will be treated as potentially compressed, and the compress bits will be looked up for every ROP tile in the surface
reserved9	fixed<0,5,0> (5 bits)	187	
DecompressCtbCount	fixed<0,8,0> (8 bits)	192	The number of CompTagBuffer (CTB) entries that are assigned to this slot. Software has to allocate the total available compatg entries to each of the slots, based on the scaling ratios etc.
DecompressZbcColor	fixed<0,32,0> (32 bits)	200	The Zero-Bandwidth-Clear color to be used when a ROP tile is ZBC compressed.
reserved12	fixed<0,24,0> (24 bits)	232	
SourceRectLeft	fixed<0,30,0> (30 bits)	256	The source rectangle defines the region of pixels that will be read from the source surface. Any pixel data outside of this will not be used inside VIC but might still be read. The source rectangle co-ordinates are specified in a U14.4 format, allowing us to specify fractional co-ordinates. The source rectangle should lie entirely within the input surface for the slot, that is, negative values, or values greater than the size of the surface are illegal.
reserved14	fixed<0,2,0> (2 bits)	286	
SourceRectRight	fixed<0,30,0> (30 bits)	288	
reserved15	fixed<0,2,0> (2 bits)	318	
SourceRectTop	fixed<0,30,0> (30 bits)	320	
reserved16	fixed<0,2,0> (2 bits)	350	
SourceRectBottom	fixed<0,30,0> (30 bits)	352	
reserved17	fixed<0,2,0> (2 bits)	382	
DestRectLeft	fixed<0,14,0> (14 bits)	384	The destination rectangle defines the region of the output surface that is affected by this slot. Together with the source rectangle it also defines the scaling rations. Any pixel outside of this region will not be affected by this input stream. For any non-4:4:4 format all corners need to fall on a multiple of 2 (Right and Bottom are minus 1 encoded though).
reserved18	fixed<0,2,0> (2 bits)	398	
DestRectRight	fixed<0,14,0> (14 bits)	400	
reserved19	fixed<0,2,0> (2 bits)	414	
DestRectTop	fixed<0,14,0> (14 bits)	416	
reserved20	fixed<0,2,0> (2 bits)	430	
DestRectBottom	fixed<0,14,0> (14 bits)	432	
reserved21	fixed<0,2,0> (2 bits)	446	

Table 74: SlotConfig Details

Name	Data Type	Offset	Description
reserved22	fixed<0,32,0> (32 bits)	448	
reserved23	fixed<0,32,0> (32 bits)	480	

SlotSurfaceConfig

Surface parameters for each slot

Table 75: SlotSurfaceConfig Details

Name	Data Type	Offset	Description
SlotPixelFormat	fixed<0,7,0> (7 bits)	0	Pixel format for each stream (PIXEL_FORMAT) Only NV12, YUY2, and UYVY allow for motion buffer calculation and DiSi1/DiNewBob de-interlacing and noise reduction.
SlotChromaLocHoriz	fixed<0,2,0> (2 bits)	7	Horizontal chroma location - (0: co-located with even luma; 1: in between even and odd luma; 2: co-located with odd luma; 3: between odd and next even luma;)
SlotChromaLocVert	fixed<0,2,0> (2 bits)	9	Vertical chroma location - (0: co-located with even luma; 1: in between even and odd luma; 2: co-located with odd luma; 3: between odd and next even luma)
SlotBlkKind	fixed<0,4,0> (4 bits)	11	Block-linear kind (BLK_KIND)
SlotBlkHeight	fixed<0,4,0> (4 bits)	15	Block-linear height (in gobs, log2)
SlotCacheWidth	fixed<0,3,0> (3 bits)	19	Number of horizontal bytes per surface-cache cache-line. Each 256B cache line can store a source region of size as enumerated below. 0: 16Bx16 (BL16Bx2) 1: 32Bx8 (BL16Bx2) 2: 64Bx4 (BL16Bx2, PL) 3: 128Bx2 (BL16Bx2, PL) 4: 256Bx1 (PL)
reserved0	fixed<0,10,0> (10 bits)	22	
SlotSurfaceWidth	fixed<0,14,0> (14 bits)	32	Width of surface minus 1. Any pixel data outside of this will not be used inside VIC but might still be read.
SlotSurfaceHeight	fixed<0,14,0> (14 bits)	46	Height of surface minus 1. Any pixel data outside of this will not be used inside VIC but might still be read.
reserved1	fixed<0,4,0> (4 bits)	60	
SlotLumaWidth	fixed<0,14,0> (14 bits)	64	Padded luma width of surface minus 1. Any pixel data outside of this will not be read.
SlotLumaHeight	fixed<0,14,0> (14 bits)	78	Padded luma height of surface minus 1. Any pixel data outside of this will not be read.
reserved2	fixed<0,4,0> (4 bits)	92	
SlotChromaWidth	fixed<0,14,0> (14 bits)	96	Padded chroma width of surface minus 1. Any pixel data outside of this will not be read. This value is not required for pixel interleaved surfaces such as ARGB
SlotChromaHeight	fixed<0,14,0> (14 bits)	110	Padded chroma height of surface minus 1. Any pixel data outside of this will not be read. This value is not required for pixel interleaved surfaces such as ARGB
reserved3	fixed<0,4,0> (4 bits)	124	

LumaKeyStruct

Used to enable Luma keying and plane alpha parameters.

Table 76: LumaKeyStruct Details

Name	Data Type	Offset	Description
luma_coeff0	fixed<0,20,0> (20 bits)	0	Matrix entry (0) of 4x1 luma conversion matrix in S12.8 format
luma_coeff1	fixed<0,20,0> (20 bits)	20	Matrix entry (1) of 4x1 luma conversion matrix in S12.8 format
luma_coeff2	fixed<0,20,0> (20 bits)	40	Matrix entry (2) of 4x1 luma conversion matrix in S12.8 format
luma_r_shift	fixed<0,4,0> (4 bits)	60	The result of the matrix multiplication will be right shifted by r_shift. To allow for highest accuracy, r_shift should always be as high as possible without losing any range in the other coefficients.
luma_coeff3	fixed<0,20,0> (20 bits)	64	Matrix entry (3) of 4x1 luma conversion matrix in S12.8 format. Is not affected by r_shift.
LumaKeyLower	fixed<0,10,0> (10 bits)	84	Lower luma key value
LumaKeyUpper	fixed<0,10,0> (10 bits)	94	Upper luma key value
LumaKeyEnabled	fixed<0,1,0> (1 bit)	104	Luma key enable
reserved0	fixed<0,2,0> (2 bits)	105	
reserved1	fixed<0,21,0> (21 bits)	107	

MatrixStruct

The matrices defined in these structures are used to specify the color space conversion between input and output pixel formats if any.

Table 77: MatrixStruct Details

Name	Data Type	Offset	Description
matrix_coeff00	fixed<0,20,0> (20 bits)	0	Matrix entry (0,0) of 4x3 color conversion matrix. Precision and right shift are the same as for the luma vector.
matrix_coeff10	fixed<0,20,0> (20 bits)	20	Matrix entry (1,0) of 4x3 color conversion matrix
matrix_coeff20	fixed<0,20,0> (20 bits)	40	Matrix entry (2,0) of 4x3 color conversion matrix
matrix_r_shift	fixed<0,4,0> (4 bits)	60	Right shift value for matrix
matrix_coeff01	fixed<0,20,0> (20 bits)	64	Matrix entry (0,1) of 4x3 color conversion matrix
matrix_coeff11	fixed<0,20,0> (20 bits)	84	Matrix entry (1,1) of 4x3 color conversion matrix
matrix_coeff21	fixed<0,20,0> (20 bits)	104	Matrix entry (2,1) of 4x3 color conversion matrix
reserved0	fixed<0,3,0> (3 bits)	124	
matrix_enable	fixed<0,1,0> (1 bit)	127	
matrix_coeff02	fixed<0,20,0> (20 bits)	128	Matrix entry (0,2) of 4x3 color conversion matrix

Table 77: MatrixStruct Details

Name	Data Type	Offset	Description
matrix_coeff12	fixed<0,20,0> (20 bits)	148	Matrix entry (1,2) of 4x3 color conversion matrix
matrix_coeff22	fixed<0,20,0> (20 bits)	168	Matrix entry (2,2) of 4x3 color conversion matrix
reserved1	fixed<0,4,0> (4 bits)	188	
matrix_coeff03	fixed<0,20,0> (20 bits)	192	Matrix entry (0,3) of 4x3 color conversion matrix
matrix_coeff13	fixed<0,20,0> (20 bits)	212	Matrix entry (1,3) of 4x3 color conversion matrix
matrix_coeff23	fixed<0,20,0> (20 bits)	232	Matrix entry (2,3) of 4x3 color conversion matrix
reserved2	fixed<0,4,0> (4 bits)	252	

BlendingSlotStruct

Input parameters to blend equations in DXVAHD_ALPHA_FILL_MODE_COMPOSITED alpha fill mode

- outputR = (srcFactC * srcR) + (dstFactC * dstR)
- outputG = (srcFactC * srcG) + (dstFactC * dstG)
- outputB = (srcFactC * srcB) + (dstFactC * dstB)
- outputA = (srcFactA * srcA) + (dstFactA * dstA)

Table 78: BlendingSlotStruct Details

Name	Data Type	Offset	Description
AlphaK1	fixed<0,10,0> (10 bits)	0	Constant Alpha value
reserved0	fixed<0,6,0> (6 bits)	10	Constant Alpha value
AlphaK2	fixed<0,10,0> (10 bits)	16	Constant Alpha value
reserved1	fixed<0,6,0> (6 bits)	26	
SrcFactCMatchSelect	fixed<0,3,0> (3 bits)	32	Blend Source Factor for Color if the color key comparison matches (BLEND_SRCFACTC)
reserved2	fixed<0,1,0> (1 bit)	35	
DstFactCMatchSelect	fixed<0,3,0> (3 bits)	36	Blend Destination Factor for Color if the color key comparison matches (BLEND_DSTFACTC)
reserved3	fixed<0,1,0> (1 bit)	39	
SrcFactAMatchSelect	fixed<0,3,0> (3 bits)	40	Blend Source Factor for Alpha if the color key comparison matches (BLEND_SRCFACTA)
reserved4	fixed<0,1,0> (1 bit)	43	
DstFactAMatchSelect	fixed<0,3,0> (3 bits)	44	Blend Source Factor for Alpha if the color key comparison matches (BLEND_DSTFACTA)
reserved5	fixed<0,1,0> (1 bit)	47	
reserved6	fixed<0,4,0> (4 bits)	48	

Table 78: BlendingSlotStruct Details

Name	Data Type	Offset	Description
reserved7	fixed<0,4,0> (4 bits)	52	
reserved8	fixed<0,4,0> (4 bits)	56	
reserved9	fixed<0,4,0> (4 bits)	60	
reserved10	fixed<0,2,0> (2 bits)	64	
OverrideR	fixed<0,10,0> (10 bits)	66	Source RGBA override values
OverrideG	fixed<0,10,0> (10 bits)	76	
OverrideB	fixed<0,10,0> (10 bits)	86	
OverrideA	fixed<0,10,0> (10 bits)	96	
reserved11	fixed<0,2,0> (2 bits)	106	
UseOverrideR	fixed<0,1,0> (1 bit)	108	Enable source RGBA value override
UseOverrideG	fixed<0,1,0> (1 bit)	109	
UseOverrideB	fixed<0,1,0> (1 bit)	110	
UseOverrideA	fixed<0,1,0> (1 bit)	111	
MaskR	fixed<0,1,0> (1 bit)	112	Per-component blend output override enables (replace blend output with destination RGBA values)
MaskG	fixed<0,1,0> (1 bit)	113	
MaskB	fixed<0,1,0> (1 bit)	114	
MaskA	fixed<0,1,0> (1 bit)	115	
reserved12	fixed<0,12,0> (12 bits)	116	

15.4.3.2 Palette Structure

PaletteStruct sends palette information for indexed formats. Each slot has its own space in the palette buffer even if the format is not indexed. For non-indexed formats the space should be left blank and should not be used by any other slots to send palette data. The number of palette entries for each slot can be at most 256.

15.4.3.3 History Buffer

The History buffer will be used to store inter-frame communication data. The structure holds cadence related information for each slot.

First dword of the hist_control (control_vector) should be set by the driver and informs Falcon whether to calculate the LUT for histogram enhancement and to calculate cadence or not.

If histogram enhancement is enabled for a slot and FALCON_CONTROL is set (explained in the SetControlParams() method) then Falcon calculates the LUT and updates the HistogramLutStruct's multiplier table of the config structure. If histogram

enhancement is enabled and FALCON_CONTROL is 0, then driver should calculate the LUT. And if the histogram enhancement flag is 0, then the table should be initialized to 1024.

Second part of the history buffer is used to store the 64 bin histogram data. This data is produced by the VIC and is used by either Falcon or driver to calculate the LUT.

15.4.3.4 FCE Microcode Buffer

This buffer holds the Fetch Control Engine (FCE) microcode that Falcon should load to the FCE unit. The driver should also pass the size of the FCE microcode with the SetFceUcodeSize() method.

15.4.3.5 Input Picture Buffers

Each picture is a separate buffer with its own surface offset method. The driver needs to send these offset methods for only the used pictures (source and reference) for the current execute.

15.4.3.6 Noise-Reduced Picture Buffers

This buffer is used by the app to keep a noise-reduced picture data of future fields if DeNoise is enabled. In the next execute, noise-reduced picture data is used as the current field, and noise-reduced picture data of the previous execute will be used as previous picture. Noise reduction requires a forward and a backward reference field for interlaced content and a backward reference for progressive content.

In the NV24 case, the VIC app uses an additional surface to keep noise reduced field of the current picture. In this case, the current noise reduced surfaces are used as previous reference surfaces.

Noise-reduced picture buffer dimensions for a slot should be same as input buffer of that slot.

15.4.3.7 Motion Map Buffers

These buffers keep motion map data of the current and previous fields if MotionMap is enabled. The previous motion map buffer should be initialized to 255 (both luma and chroma) at the start of the clip. Use the motion map buffer of the previous execute pass prev motion map in the current execute for subsequent frames. MotionMap calculation is required for DiSi1 and also needs a forward and a backward reference. Behavior is not defined for progressive streams.

Motion map buffer dimensions for a slot should be same as input buffer of that slot.

15.4.3.8 Output Buffer

This buffer stores output data.

15.4.3.9 CRC Buffer

The InterfaceCrcStruct and InputCrcStruct are shared to store the CRC information. These are selected based on method SetCrcMode.

15.4.4 Application Error Codes

The following table defines the error codes.

Table 79: Error Codes

Name	Value	Description
MSDECOS_ERROR_NONE	0x00000000	Default return code for app
MSDECOS_ERROR_EXECUTE_INSUFFICIENT_DATA	0x00000001	To be returned by the app to the OS
MSDECOS_ERROR_SEMAPHORE_INSUFFICIENT_DATA	0x00000002	Insufficient semaphore methods received. Method Data written out into MAILBOX1. MAILBOX0 bits 31:20- Method ID. MAILBOX0 bits 19-12: Method Count (SWFIFO + MTHDCOUNT register)
MSDECOS_ERROR_INVALID_METHOD	0x00000003	Unsupported method

Table 79: Error Codes

Name	Value	Description
MSDECOS_ERROR_INVALID_DMA_PAGE	0x00000004	Unsupported method
MSDECOS_ERROR_UNHANDLED_INTERRUPT	0x00000005	The app has either no interrupt handler or an unhandled OS error
MSDECOS_ERROR_EXCEPTION	0x00000006	Exception raised by Falcon
MSDECOS_ERROR_INVALID_CTXSW_REQUEST	0x00000007	Invalid CTXSW request to the OS
MSDECOS_ERROR_APPLICATION	0x00000008	Application returned a nonzero error code
MSDECOS_ERROR_SWBREAKPT	0x00000009	Exception raised to dump registers in debug mode
VicErrorConfigStructSizeMismatch	0xb0b60060	The size of the config structure as defined in the SetControlParams method does not match the expected size of the ConfigStructure for the current VIC hardware
VicErrorInvalidInputFormatSlot_0	0xb0b60070	The input pixel format specified for slot 0 does not match one of the supported input pixel formats
VicErrorInvalidInputFormatSlot_1	0xb0b60071	The input pixel format specified for slot 1 does not match one of the supported input pixel formats
VicErrorInvalidInputFormatSlot_2	0xb0b60072	The input pixel format specified for slot 2 does not match one of the supported input pixel formats
VicErrorInvalidInputFormatSlot_3	0xb0b60073	The input pixel format specified for slot 3 does not match one of the supported input pixel formats
VicErrorInvalidInputFormatSlot_4	0xb0b60074	The input pixel format specified for slot 4 does not match one of the supported input pixel formats
VicErrorInvalidInputFormatSlot_5	0xb0b60075	The input pixel format specified for slot 5 does not match one of the supported input pixel formats
VicErrorInvalidInputFormatSlot_6	0xb0b60076	The input pixel format specified for slot 6 does not match one of the supported input pixel formats
VicErrorInvalidInputFormatSlot_7	0xb0b60077	The input pixel format specified for slot 7 does not match one of the supported input pixel formats
VicErrorInvalidOutputFormat	0xb0b60080	The output pixel format specified does not match one of the supported output pixel formats
VicErrorApptimerExpired	0xb0b60090	The Falcon microcode is taking longer than expected to complete processing the current frame
VicErrorFcePanicErrIntr	0xb0b600a0	The fetch-control engine threw a panic. Usually indicates a bug in the FCE microcode.
VicErrorFceDivisionByZeroErrIntr	0xb0b600a5	The fetch-control engine threw a divide-by-zero error. Can be caused either by a bug in the FCE microcode, or invalid inputs in the ConfigStruct rectangle specifications.
VicErrorScBankConflictErrIntr	0xb0b600a1	There was a bank conflict error in the surface cache. This can only be caused if there is a bug in either the FetchControl or SurfaceCache hardware
VicErrorScResetErrIntr	0xb0b600a2	SurfaceCache was reset while there were still locked entries in the cache
VicErrorYsSkipdErrIntr	0xb0b600a3	There was a problem in the FetchControl calculation of filter skips for the Y-scaler
VicErrorXsSkipdErrIntr	0xb0b600a4	There was a problem in the FetchControl calculation of filter skips for the X-scaler

15.4.5 Application Method

The VIC application method registers are accessed directly by the driver. Because the VIC uses the shared MSDEC Falcon OS, two ranges of methods are used (in the MSDEC, there are common and application specific ranges). The range of method registers is:

- 0x700-0x77F (common method range) and
- 0x400-0x6FF (method range corresponding to overlay 1)

The following table defines the method map.

Table 80: Method Map

Address	Method
0x0100	VIC.Nop
0x0140	VIC.PmTrigger
0x0200	VIC.SetApplicationID
0x0204	VIC.SetWatchdogTimer
0x0240	VIC.SemaphoreA
0x0244	VIC.SemaphoreB
0x0248	VIC.SemaphoreC
0x024c	VIC.CtxSaveArea
0x0250	VIC.CtxSwitch
0x0300	VIC.Execute
0x0304	VIC.SemaphoreD
0x0400	VIC.SetSurface0Slot0LumaOffset[0]
0x0404	VIC.SetSurface0Slot0ChromaU_Offset[0]
0x0408	VIC.SetSurface0Slot0ChromaV_Offset[0]
0x040c	VIC.SetSurface1Slot0LumaOffset[0]
0x0410	VIC.SetSurface1Slot0ChromaU_Offset
0x0414	VIC.SetSurface1Slot0ChromaV_Offset[0]
0x0418	VIC.SetSurface2Slot0LumaOffset[0]
0x041c	VIC.SetSurface2Slot0ChromaU_Offset[0]
0x0420	VIC.SetSurface2Slot0ChromaV_Offset[0]
0x0424	VIC.SetSurface3Slot0LumaOffset[0]
0x0428	VIC.SetSurface3Slot0ChromaU_Offset[0]
0x042c	VIC.SetSurface3Slot0ChromaV_Offset[0]
0x0430	VIC.SetSurface4Slot0LumaOffset[0]
0x0434	VIC.SetSurface4Slot0ChromaU_Offset[0]
0x0438	VIC.SetSurface4Slot0ChromaV_Offset[0]
0x043c	VIC.SetSurface5Slot0LumaOffset[0]
0x0440	VIC.SetSurface5Slot0ChromaU_Offset[0]
0x0444	VIC.SetSurface5Slot0ChromaV_Offset[0]
0x0448	VIC.SetSurface6Slot0LumaOffset[0]
0x044c	VIC.SetSurface6Slot0ChromaU_Offset[0]
0x0450	VIC.SetSurface6Slot0ChromaV_Offset[0]
0x0454	VIC.SetSurface7Slot0LumaOffset[0]
0x0458	VIC.SetSurface7Slot0ChromaU_Offset[0]
0x045c	VIC.SetSurface7Slot0ChromaV_Offset[0]
0x0460	VIC.SetSurface0Slot1LumaOffset[1]
0x0464	VIC.SetSurface0Slot1ChromaU_Offset[1]
0x0468	VIC.SetSurface0Slot1ChromaV_Offset[1]

Table 80: Method Map

Address	Method
0x046c	VIC.SetSurface1Slot1LumaOffset[1]
0x0470	VIC.SetSurface1Slot1ChromaU_Offset[1]
0x0474	VIC.SetSurface1Slot1ChromaV_Offset[1]
0x0478	VIC.SetSurface2Slot1LumaOffset[1]
0x047c	VIC.SetSurface2Slot1ChromaU_Offset[1]
0x0480	VIC.SetSurface2Slot1ChromaV_Offset[1]
0x0484	VIC.SetSurface3Slot1LumaOffset[1]
0x0488	VIC.SetSurface3Slot1ChromaU_Offset[1]
0x048c	VIC.SetSurface3Slot1ChromaV_Offset[1]
0x0490	VIC.SetSurface4Slot1LumaOffset[1]
0x0494	VIC.SetSurface4Slot1ChromaU_Offset[1]
0x0498	VIC.SetSurface4Slot1ChromaV_Offset[1]
0x049c	VIC.SetSurface5Slot1LumaOffset[1]
0x04a0	VIC.SetSurface5Slot1ChromaU_Offset[1]
0x04a4	VIC.SetSurface5Slot1ChromaV_Offset[1]
0x04a8	VIC.SetSurface6Slot1LumaOffset[1]
0x04ac	VIC.SetSurface6Slot1ChromaU_Offset[1]
0x04b0	VIC.SetSurface6Slot1ChromaV_Offset[1]
0x04b4	VIC.SetSurface7Slot1LumaOffset[1]
0x04b8	VIC.SetSurface7Slot1ChromaU_Offset[1]
0x04bc	VIC.SetSurface7Slot1ChromaV_Offset[1]
0x04c0	VIC.SetSurface0Slot2LumaOffset[2]
0x04c4	VIC.SetSurface0Slot2ChromaU_Offset[2]
0x04c8	VIC.SetSurface0Slot2ChromaV_Offset[2]
0x04cc	VIC.SetSurface1Slot2LumaOffset[2]
0x04d0	VIC.SetSurface1Slot2ChromaU_Offset[2]
0x04d4	VIC.SetSurface1Slot2ChromaV_Offset[2]
0x04d8	VIC.SetSurface2Slot2LumaOffset[2]
0x04dc	VIC.SetSurface2Slot2ChromaU_Offset[2]
0x04e0	VIC.SetSurface2Slot2ChromaV_Offset[2]
0x04e4	VIC.SetSurface3Slot2LumaOffset[2]
0x04e8	VIC.SetSurface3Slot2ChromaU_Offset[2]
0x04ec	VIC.SetSurface3Slot2ChromaV_Offset[2]
0x04f0	VIC.SetSurface4Slot2LumaOffset[2]
0x04f4	VIC.SetSurface4Slot2ChromaU_Offset[2]
0x04f8	VIC.SetSurface4Slot2ChromaV_Offset[2]
0x04fc	VIC.SetSurface5Slot2LumaOffset[2]
0x0500	VIC.SetSurface5Slot2ChromaU_Offset[2]
0x0504	VIC.SetSurface5Slot2ChromaV_Offset[2]
0x0508	VIC.SetSurface6Slot2LumaOffset[2]
0x050c	VIC.SetSurface6Slot2ChromaU_Offset[2]
0x0510	VIC.SetSurface6Slot2ChromaV_Offset[2]
0x0514	VIC.SetSurface7Slot2LumaOffset[2]
0x0518	VIC.SetSurface7Slot2ChromaU_Offset[2]
0x051c	VIC.SetSurface7Slot2ChromaV_Offset[2]
0x0520	VIC.SetSurface0Slot3LumaOffset[3]

Table 80: Method Map

Address	Method
0x0524	VIC.SetSurface0Slot3ChromaU_Offset[3]
0x0528	VIC.SetSurface0Slot3ChromaV_Offset[3]
0x052c	VIC.SetSurface1Slot3LumaOffset[3]
0x0530	VIC.SetSurface1Slot3ChromaU_Offset[3]
0x0534	VIC.SetSurface1Slot3ChromaV_Offset[3]
0x0538	VIC.SetSurface2Slot3LumaOffset[3]
0x053c	VIC.SetSurface2Slot3ChromaU_Offset[3]
0x0540	VIC.SetSurface2Slot3ChromaV_Offset[3]
0x0544	VIC.SetSurface3Slot3LumaOffset[3]
0x0548	VIC.SetSurface3Slot3ChromaU_Offset[3]
0x054c	VIC.SetSurface3Slot3ChromaV_Offset[3]
0x0550	VIC.SetSurface4Slot3LumaOffset[3]
0x0554	VIC.SetSurface4Slot3ChromaU_Offset[3]
0x0558	VIC.SetSurface4Slot3ChromaV_Offset[3]
0x055c	VIC.SetSurface5Slot3LumaOffset[3]
0x0560	VIC.SetSurface5Slot3ChromaU_Offset[3]
0x0564	VIC.SetSurface5Slot3ChromaV_Offset[3]
0x0568	VIC.SetSurface6Slot3LumaOffset[3]
0x056c	VIC.SetSurface6Slot3ChromaU_Offset[3]
0x0570	VIC.SetSurface6Slot3ChromaV_Offset[3]
0x0574	VIC.SetSurface7Slot3LumaOffset[3]
0x0578	VIC.SetSurface7Slot3ChromaU_Offset[3]
0x057c	VIC.SetSurface7Slot3ChromaV_Offset[3]
0x0580	VIC.SetSurface0Slot4LumaOffset[4]
0x0584	VIC.SetSurface0Slot4ChromaU_Offset[4]
0x0588	VIC.SetSurface0Slot4ChromaV_Offset[4]
0x058c	VIC.SetSurface1Slot4LumaOffset[4]
0x0590	VIC.SetSurface1Slot4ChromaU_Offset[4]
0x0594	VIC.SetSurface1Slot4ChromaV_Offset[4]
0x0598	VIC.SetSurface2Slot4LumaOffset[4]
0x059c	VIC.SetSurface2Slot4ChromaU_Offset[4]
0x05a0	VIC.SetSurface2Slot4ChromaV_Offset[4]
0x05a4	VIC.SetSurface3Slot4LumaOffset[4]
0x05a8	VIC.SetSurface3Slot4ChromaU_Offset[4]
0x05ac	VIC.SetSurface3Slot4ChromaV_Offset[4]
0x05b0	VIC.SetSurface4Slot4LumaOffset[4]
0x05b4	VIC.SetSurface4Slot4ChromaU_Offset[4]
0x05b8	VIC.SetSurface4Slot4ChromaV_Offset[4]
0x05bc	VIC.SetSurface5Slot4LumaOffset[4]
0x05c0	VIC.SetSurface5Slot4ChromaU_Offset[4]
0x05c4	VIC.SetSurface5Slot4ChromaV_Offset[4]
0x05c8	VIC.SetSurface6Slot4LumaOffset[4]
0x05cc	VIC.SetSurface6Slot4ChromaU_Offset[4]
0x05d0	VIC.SetSurface6Slot4ChromaV_Offset[4]
0x05d4	VIC.SetSurface7Slot4LumaOffset[4]
0x05d8	VIC.SetSurface7Slot4ChromaU_Offset[4]

Table 80: Method Map

Address	Method
0x05dc	VIC.SetSurface7Slot4ChromaV_Offset[4]
0x05e0	VIC.SetSurface0LumaOffset[5]
0x05e4	VIC.SetSurface0ChromaU_Offset[5]
0x05e8	VIC.SetSurface0ChromaV_Offset[5]
0x05ec	VIC.SetSurface1LumaOffset[5]
0x05f0	VIC.SetSurface1ChromaU_Offset[5]
0x05f4	VIC.SetSurface1ChromaV_Offset[5]
0x05f8	VIC.SetSurface2LumaOffset[5]
0x05fc	VIC.SetSurface2ChromaU_Offset[5]
0x0600	VIC.SetSurface2ChromaV_Offset[5]
0x0604	VIC.SetSurface3LumaOffset[5]
0x0608	VIC.SetSurface3ChromaU_Offset[5]
0x060c	VIC.SetSurface3ChromaV_Offset[5]
0x0610	VIC.SetSurface4LumaOffset[5]
0x0614	VIC.SetSurface4ChromaU_Offset[5]
0x0618	VIC.SetSurface4ChromaV_Offset[5]
0x061c	VIC.SetSurface5LumaOffset[5]
0x0620	VIC.SetSurface5ChromaU_Offset[5]
0x0624	VIC.SetSurface5ChromaV_Offset[5]
0x0628	VIC.SetSurface6LumaOffset[5]
0x062c	VIC.SetSurface6ChromaU_Offset[5]
0x0630	VIC.SetSurface6ChromaV_Offset[5]
0x0634	VIC.SetSurface7LumaOffset[5]
0x0638	VIC.SetSurface7ChromaU_Offset[5]
0x063c	VIC.SetSurface7ChromaV_Offset[5]
0x0640	VIC.SetSurface0LumaOffset[6]
0x0644	VIC.SetSurface0ChromaU_Offset[6]
0x0648	VIC.SetSurface0ChromaV_Offset[6]
0x064c	VIC.SetSurface1LumaOffset[6]
0x0650	VIC.SetSurface1ChromaU_Offset[6]
0x0654	VIC.SetSurface1ChromaV_Offset[6]
0x0658	VIC.SetSurface2LumaOffset[6]
0x065c	VIC.SetSurface2ChromaU_Offset[6]
0x0660	VIC.SetSurface2ChromaV_Offset[6]
0x0664	VIC.SetSurface3LumaOffset[6]
0x0668	VIC.SetSurface3ChromaU_Offset[6]
0x066c	VIC.SetSurface3ChromaV_Offset[6]
0x0670	VIC.SetSurface4LumaOffset[6]
0x0674	VIC.SetSurface4ChromaU_Offset[6]
0x0678	VIC.SetSurface4ChromaV_Offset[6]
0x067c	VIC.SetSurface5LumaOffset[6]
0x0680	VIC.SetSurface5ChromaU_Offset[6]
0x0684	VIC.SetSurface5ChromaV_Offset[6]
0x0688	VIC.SetSurface6LumaOffset[6]
0x068c	VIC.SetSurface6ChromaU_Offset[6]
0x0690	VIC.SetSurface6ChromaV_Offset[6]

Table 80: Method Map

Address	Method
0x0694	VIC.SetSurface7LumaOffset[6]
0x0698	VIC.SetSurface7ChromaU_Offset[6]
0x069c	VIC.SetSurface7ChromaV_Offset[6]
0x06a0	VIC.SetSurface0LumaOffset[7]
0x06a4	VIC.SetSurface0ChromaU_Offset[7]
0x06a8	VIC.SetSurface0ChromaV_Offset[7]
0x06ac	VIC.SetSurface1LumaOffset[7]
0x06b0	VIC.SetSurface1ChromaU_Offset[7]
0x06b4	VIC.SetSurface1ChromaV_Offset[7]
0x06b8	VIC.SetSurface2LumaOffset[7]
0x06bc	VIC.SetSurface2ChromaU_Offset[7]
0x06c0	VIC.SetSurface2ChromaV_Offset[7]
0x06c4	VIC.SetSurface3LumaOffset[7]
0x06c8	VIC.SetSurface3ChromaU_Offset[7]
0x06cc	VIC.SetSurface3ChromaV_Offset[7]
0x06d0	VIC.SetSurface4LumaOffset[7]
0x06d4	VIC.SetSurface4ChromaU_Offset[7]
0x06d8	VIC.SetSurface4ChromaV_Offset[7]
0x06dc	VIC.SetSurface5LumaOffset[7]
0x06e0	VIC.SetSurface5ChromaU_Offset[7]
0x06e4	VIC.SetSurface5ChromaV_Offset[7]
0x06e8	VIC.SetSurface6LumaOffset[7]
0x06ec	VIC.SetSurface6ChromaU_Offset[7]
0x06f0	VIC.SetSurface6ChromaV_Offset[7]
0x06f4	VIC.SetSurface7LumaOffset[7]
0x06f8	VIC.SetSurface7ChromaU_Offset[7]
0x06fc	VIC.SetSurface7ChromaV_Offset[7]
0x0700	VIC.SetPictureIndex
0x0704	VIC.SetControlParams
0x0708	VIC.SetConfigStructOffset
0x070c	VIC.SetFilterStructOffset
0x0710	VIC.SetPaletteOffset
0x0714	VIC.SetHistOffset
0x0718	VIC.SetContextId
0x071c	VIC.SetFceUcodeSize
0x0720	VIC.SetOutputSurfaceLumaOffset
0x0724	VIC.SetOutputSurfaceChromaU_Offset
0x0728	VIC.SetOutputSurfaceChromaV_Offset
0x072c	VIC.SetFceUcodeOffset
0x0740	VIC.SetSlotContextId[0]
0x0744	VIC.SetSlotContextId[1]
0x0748	VIC.SetSlotContextId[2]
0x074c	VIC.SetSlotContextId[3]
0x0750	VIC.SetSlotContextId[4]
0x0754	VIC.SetSlotContextId[5]
0x0758	VIC.SetSlotContextId[6]

Table 80: Method Map

Address	Method
0x075c	VIC.SetSlotContextId[7]
0x0760	VIC.SetCompTagBuffer_Offset[0]
0x0764	VIC.SetCompTagBuffer_Offset[1]
0x0768	VIC.SetCompTagBuffer_Offset[2]
0x076c	VIC.SetCompTagBuffer_Offset[3]
0x0770	VIC.SetCompTagBuffer_Offset[4]
0x0774	VIC.SetCompTagBuffer_Offset[5]
0x0778	VIC.SetCompTagBuffer_Offset[6]
0x077c	VIC.SetCompTagBuffer_Offset[7]
0x0780	VIC.SetHistoryBufferOffset[0]
0x0784	VIC.SetHistoryBufferOffset[1]
0x0788	VIC.SetHistoryBufferOffset[2]
0x078c	VIC.SetHistoryBufferOffset[3]
0x0790	VIC.SetHistoryBufferOffset[4]
0x0794	VIC.SetHistoryBufferOffset[5]
0x0798	VIC.SetHistoryBufferOffset[6]
0x079c	VIC.SetHistoryBufferOffset[7]
0x1114	VIC.PmTriggerEnd

15.4.6 Method Naming and Programming

The VIC uses generic names to avoid confusing name overloading to the 16 surfaces (8 luma, 8 chroma) methods for the MAX_SLOTS

- SurfacexSlotyLumaOffset[y]
- SetSurfacexSlotyChromaU_Offset[y]
- SetSurfacexSlotyChromaV_Offset[y] with x in [0..7] and y in [0..MAX_SLOTS-1]

The following sets of methods are required for a given ConfigStruct setup. Any setup not covered in this section should be regarded as illegal.

Note:

- NV24 frames are treated as NV12 fields. The frame format for NV24 has to be set to FRAME_FORMAT_TOP_FIELD.
- In case of noise reduction, all previously noise reduced surfaces should be used as references instead of the original ones. In case of interlaced fields this also applies to the current fields as they have previously been noise reduced.
- Surface6 has been obsoleted and should not be used.
- History buffer is always required and should be owned by Falcon.

15.4.6.1 Application ID

Application ID is not needed as there is only one application for the engine (unlike different codecs for the MSDEC). But to use it with MSDEC infrastructure, the driver needs to pass the ApplicationId as 1 (corresponding to overlay 1).

15.4.6.2 Progressive Frames

No Preprocessing

Enable bits have to be set to 0x01 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()

Noise Reduction

Enable bits have to be set to 0x07 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference surface)
- SetSurface2SlotLumaOffset() (noise reduced surface)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference surface)
- SetSurface2SlotChromaOffset() (noise reduced surface)

15.4.6.3 Progressive Fields

No Preprocessing

Enable bits have to be set to 0x03 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset() (top field)
- SetSurface1SlotLumaOffset() (bottom field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset() (top field)
- SetSurface1SlotChromaOffset() (bottom field)

Noise Reduction

Enable bits have to be set to 0x03f in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset() (top field)
- SetSurface1SlotLumaOffset() (bottom field)
- SetSurface2SlotLumaOffset() (top field of backward reference surface)
- SetSurface3SlotLumaOffset() (bottom field of backward reference surface)
- SetSurface4SlotLumaOffset() (top field of noise reduced surface)
- SetSurface5SlotLumaOffset() (bottom field of noise reduced surface)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset() (top field)
- SetSurface1SlotChromaOffset() (bottom field)
- SetSurface2SlotChromaOffset() (top field of backward reference surface)
- SetSurface3SlotChromaOffset() (bottom field of backward reference surface)

- SetSurface4SlotChromaOffset() (top field of noise reduced surface)
- SetSurface5SlotChromaOffset() (bottom field of noise reduced surface)

15.4.6.4 Interlaced Frames

No Preprocessing

Enable bits have to be set to 0x01 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()

Noise Reduction

Enable bits have to be set to 0x07 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference surface)
- SetSurface2SlotLumaOffset() (noise reduced surface)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference surface)
- SetSurface2SlotChromaOffset() (noise reduced surface)

15.4.6.5 Interlaced Fields

No Preprocessing

This is for BOB_FIELD only. For WEAVE, see “No Preprocessing” under “Progressive Fields“.

Enable bits have to be set to 0x01 in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()

Cadence Detection

Enable bits have to be set to 0x07 in fetchControl0Struct. This also requires cadence detection to be enabled.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)

- SetSurface2SlotChromaOffset() (forward reference field)

Motion Calculation

Enable bits have to be set to 0x17 in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)

Motion Calculation and Cadence Detection

Same as above in Motion Calculation except that cadence detection is enabled.

DISI1 (Motion Calculation and Motion Combine)

Enable bits have to be set to 0xb7 in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back. The combined motion field is generated out of current motion and previous motion to be used in DiSi1 deinterlacer.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)
- SetSurface5SlotLumaOffset() (previous motion field)
- SetSurface7SlotLumaOffset() (combined motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)
- SetSurface5SlotChromaOffset() (previous motion field)
- SetSurface7SlotChromaOffset() (combined motion field)

DISI1 (Motion Calculation and Motion Combine) and Cadence Detection

Same as above in DiSi1 except that cadence detection is enabled.

Noise Reduction

Enable bits have to be set to 0x0f in fetchControl0Struct.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (previously noise reduced backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface3SlotLumaOffset() (noise reduced forward reference field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (previously noise reduced backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface3SlotChromaOffset() (noise reduced forward reference field)

Noise Reduction and Cadence Detection

Same as above in Noise Reduction except that cadence detection is enabled.

Noise Reduction and Motion Calculation

Enable bits have to be set to 0x1f in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()
- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface3SlotLumaOffset() (noise reduced forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface3SlotChromaOffset() (noise reduced forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)

Noise Reduction and Motion Calculation and Cadence Detection

Same as above in Noise Reduction and Motion Calculation except that cadence detection is enabled.

Noise Reduction and DISI1 (Motion Calculation and Motion Combine)

Enable bits have to be set to 0xbf in fetchControl0Struct. The data in the current motion field is read first and IIR filtered with the newly calculated motion before it is written back. The combined motion field is generated out of current motion and previous motion to be used in DiSi1 deinterlacer.

Required methods for all pixel formats:

- SetSurface0SlotLumaOffset()

- SetSurface1SlotLumaOffset() (backward reference field)
- SetSurface2SlotLumaOffset() (forward reference field)
- SetSurface3SlotLumaOffset() (noise reduced forward reference field)
- SetSurface4SlotLumaOffset() (current motion field)
- SetSurface5SlotLumaOffset() (previous motion field)
- SetSurface7SlotLumaOffset() (combined motion field)

Required methods for pixel formats with separate chroma plane:

- SetSurface0SlotChromaOffset()
- SetSurface1SlotChromaOffset() (backward reference field)
- SetSurface2SlotChromaOffset() (forward reference field)
- SetSurface3SlotChromaOffset() (noise reduced forward reference field)
- SetSurface4SlotChromaOffset() (current motion field)
- SetSurface5SlotChromaOffset() (previous motion field)
- SetSurface7SlotChromaOffset() (combined motion field)

Noise Reduction and DISI1 (Motion Calculation and Motion Combine) and Cadence Detection

Same as above in Noise Reduction and DiSi1 except that cadence detection is enabled.

15.4.7 Programming Restrictions

15.4.7.1 Input parameters

- When a slot is enabled in the slotConfig structure, it should have one or more surfaces enabled in the SurfaceEnable bits.
- The source and destination rectangles should be of non-zero sizes (i.e. right >= left and bottom >= top).
- Since the Luma Width/Height parameters represent padded surface sizes, the values in these parameters should be greater than or equal to the corresponding surface width/height parameters.
- For single plane surfaces like RGB surfaces, the Chroma Width/Height parameters should be programmed to zero since the Falcon microcode uses these parameters to check for the existence of multi-plane formats.
- The Chroma Width/Height parameters should be greater than or equal to the sub-sampled size of the surface width/height parameters.
- For surfaces with sub-sampled chroma
  - The source rectangle should be big enough to have at least one chroma sample within the rectangle.
  - The input surface size should be a multiple of two in the direction of the sub-sampling
- For operations involving video pre-process operations like IVTC, TNR/TNR2, DiSi1 Deinterlacing, the supported pixel formats are restricted as below:

Operations	Pixel Formats
Deinterlacing/ Cadence Detection	T_Y8___V8U8_N420, T_Y8___U8___V8_N420, T_Y8_U8__Y8_V8, T_U8_Y8__V8_Y8
TNR/TNR2	T_Y8___V8U8_N420, T_Y8___U8___V8_N420, T_Y8_U8__Y8_V8, T_U8_Y8__V8_Y8

- When the input uses interlaced surfaces, there are the following additional restrictions:
  - The source rectangle height should be a multiple of 2 to account for the surface being comprised of 2 fields.

  - If the input pixel format uses chroma sub-sampling in the vertical direction (420 or 422R), then the source rectangle height should be a multiple of 4.
- Note that the source rectangle is always specified in "frame co-ordinates". Therefore, when deinterlacing two fields of height y each, the source rectangle height should be within $2*y$.

15.4.7.2 Scaling

- When panoramic scaling is enabled
  - The destination rectangle should be at least 4 pixels wide and 4 pixels tall
  - The maximum downscaling ratio is 7:1
- For non-panoramic scaling
  - The maximum downscaling ratio is 16:1
- When enabling the filter override mode (see DownsampleVert/Horiz), only 5 slots can be enabled. This is because of the available size of the coefficient structure.

15.4.7.3 Luma Keying

- When Luma keying is enabled, the luma key range should be non-empty (i.e. lower <= upper)

15.4.7.4 Soft clamping

- The soft clamping lower and upper thresholds should be set such that the lower threshold is less than or equal to the upper threshold at all times

15.4.7.5 Deinterlacing

- The BOB deinterlacing mode is designed to work on interlaced-frame input (i.e. the source is interlaced, but the two fields are stored interleaved together in a single frame-surface.)
- The BOB_FIELD, DISI1, NEWBOB deinterlacing modes are designed to work on interlaced-field input (i.e. the fields are stored as separate surfaces.)
- The only allowed deinterlacing mode for progressive inputs is WEAVE, where it behaves as a deinterlacing bypass.
- For interlaced inputs, the WEAVE algorithm expects field surfaces and weaves them together.

Cadence Detection

- Requires field based surfaces
- Needs forward and backward reference surfaces enabled

Motion calculation

- Motion map calculation requires field based surfaces with forward and backward reference surfaces and writes out a current motion surface
- Combine motion map calculation requires field based surfaces with current and previous motion surfaces enabled and writes out a combined motion surface

15.4.7.6 Noise Reduction

- The allowed range for the IIR strength parameter is 0x0 to 0x400 (inclusive)

Field based noise reduction

- Requires forward and backward reference surfaces enabled, as well as a noise reduced output surface

Frame based noise reduction

- Requires current and backward reference surfaces enabled, as well as a noise reduced output surface

15.4.7.7 Output parameters

- The list of pixel formats supported on the output is not exactly the same as the list of input pixel formats. For example, the palette pixel formats are not supported on output.
- When output transformations are enabled, then all destination and clear rectangles should lie entirely within the target rectangle.
- Since the Luma Width/Height parameters represent padded surface sizes, the values in these parameters should be greater than or equal to the corresponding surface width/height parameters.
- The Chroma Width/Height parameters should be greater than or equal to the sub-sampled size of the surface width/height parameters.
- For surfaces with sub-sampled chroma
  - The target and destination rectangles should be big enough to have at least one chroma sample within the rectangle.
  - The output surface size should be a multiple of two in the direction of the sub-sampling

Target rectangle

- Target rectangle should lie entirely within the output surface (no negative co-ordinates, or co-ordinates outside of the output surface size).
- The target rectangle should be non-empty (i.e. right >= left and bottom >= top).

Clear rectangles

- When a clear rectangle is enabled, it should have a non-zero width and height (i.e. right >= left and bottom >= top).

Alpha Fill Mode

- For alpha fill mode _SOURCE_ALPHA, the AlphaFillSlot parameter should be set to a valid and enabled slot

15.4.8 Performance and Power Tuning Guidelines

Fully Opaque/Fully Transparent surfaces

VIC hardware cannot know when a certain slot is fully opaque or fully transparent. Thus, when for example a fully opaque slot lies over other slots, the data for the other slots is still fetched from memory and is blended with the data for the opaque slot, even though the slots below the opaque slot will not affect the output. Software should therefore program VIC to avoid the fetch of the data below the opaque surface by putting a clear rectangle over the area beneath the opaque slot, and enabling the ClearRectMask for all the slots below the opaque slot. This software enhancement will significantly reduce the VIC bandwidth requirements, and will also enhance the VIC performance for use cases involving opaque surfaces.

Smaller Target Rectangles

For cases where the displayed image does not cover all of the screen, software should optimize bandwidth by only generating/compositing the portion of the screen which actually has usable data, and let either Display (preferably), or VIC fill in the rest of the screen with a background color. This will save both read and write bandwidth in such use cases.

Memory Formats

The CacheWidth parameter in the VIC ConfigStruct defines the mapping of the 256B surface cache cache-lines to rectangular screen space. Based on our performance testing simulations, the recommendation for programming this parameter is as below

- For pitch-linear surfaces, set the CacheWidth to 64Bx4 - i.e., CacheWidth=2
- For block-linear surfaces, set the CacheWidth to 32Bx8 - i.e., CacheWidth=1

Clock Gating

VIC has 3 levels of clock gating, just like most other units in Tegra. Software should ensure that 1st and 2nd level clock gating is enabled for VIC at all times so that the VIC hardware can try to turn off clock branches whenever any portion of the logic is unused.

- 2nd level clock gating control: Bit 5 of CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRE_0 should be set to 0.
- NV_PVIC_THI_SLCG_OVERRIDE_HIGH_A and NV_PVIC_THI_SLCG_OVERRIDE_LOW_A should both be set to 0x0 to enable automatic clock-gating of VIC sub-units when idle.

Power Gating

The VIC partition(s) can be power-gated when not in use, and software should put VIC in power gate mode for any use case that does not make use of VIC. Software should follow the power gating/ungating in order to put VIC in PG mode, or take VIC out of PG mode.

15.5 VIC THI Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

15.5.1 NV_PVIC_THI_INCR_SYNCPT

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:8	R/W	0	NV_PVIC_THI_INCR_SYNCPT_COND: 0: COND_IMMEDIATE (default) 1: COND_OP_DONE Indicates sync point condition at which THI has to return the index value back to Host1x. Following are the values of sync point conditions supported in THI: 8'h0: INCR_SYNCPT_0_COND_IMMEDIATE 8'h1: INCR_SYNCPT_0_COND_OP_DONE
7:0	R/W	0	NV_PVIC_THI_INCR_SYNCPT_INDX: 0: INDX_INIT (default) Indicates the sync point index value, THI will return this index value back to Host1x when the particular sync point condition is done.

15.5.2 NV_PVIC_THI_INCR_SYNCPT_ERR

COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:4	R	0	Reserved
1	RW1C	0	NV_PVIC_THI_INCR_SYNCPT_ERR_COND_STS_OPDONE: 0: COND_STS_OPDONE_INIT (default) 1: COND_STS_OPDONE_CLEAR
0	RW1C	0	NV_PVIC_THI_INCR_SYNCPT_ERR_COND_STS_IMM: 0: COND_STS_IMM_INIT (default) 1: COND_STS_IMM_CLEAR

15.5.3 NV_PVIC_THI_CTXSW_INCR_SYNCPT

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:8	R	0	Reserved

Bit	R/W	Reset	Description
7:0	R/W	0	NV_PVIC_THI_CTXSW_INCR_SYNCPT_INDX: 0: INDX_INIT (default) Indicates the sync point index value, THI will return this index value back to Host1x when the CONTEXT SAVE/Restore operation is done.

15.5.4 NV_PVIC_THI_CTXSW

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledgment will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts. Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they are always current.

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000F800

Bit	R/W	Reset	Description
31:28	R/W	0	NV_PVIC_THI_CTXSW_NEXT_CHANNEL: 0: NEXT_CHANNEL_INIT (default) Indicates next requested channel of engine.
27:26	R	0	Reserved
25:16	R/W	0	NV_PVIC_THI_CTXSW_NEXT_CLASS: 0: NEXT_CLASS_INIT (default) Indicates next requested class of engine.
15:12	R/W	Fh	NV_PVIC_THI_CTXSW_CURR_CHANNEL: Fh: CURR_CHANNEL_INIT (default) Indicates current working channel of engine. Reset to invalid.
11	R	1	NV_PVIC_THI_CTXSW_AUTO_ACK: 1: AUTO_ACK_INIT (default) This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they are always current.
10	R	0	Reserved
9:0	R/W	0	NV_PVIC_THI_CTXSW_CURR_CLASS: 0: CURR_CLASS_INIT (default) Indicates current working class of engine.

15.5.5 NV_PVIC_THI_CONT_SYNCPT_EOF

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	0	NV_PVIC_THI_CONT_SYNCPT_EOF_COND: 0: COND_INIT (default) Sync Point Condition Control: This bit can be used to enable/disable generation of continuous sync point increment. 1: Enable 0: Disable

Bit	R/W	Reset	Description
7:0	R/W	0	NV_PVIC_THI_CONT_SYNCPT_EOF_INDEX: 0: INDEX_INIT (default) Sync Point Counter Index: This parameter specifies the index of the sync point counter that are returned to host when continuous sync point is enabled (COND = ENABLE) and whenever an FRAME_DONE happens.

15.5.6 NV_PVIC_THI_METHOD0

There are two method registers METHOD0 and METHOD1

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:12	R	0	Reserved
11:0	R/W	0	NV_PVIC_THI_METHOD0_OFFSET: 0X 0: OFFSET_INIT (default) This contains method ID which is to be sent to Falcon over method interface. THI waits for write to METHOD_DATA (0X011) register before triggering any write over Falcon method interface.

15.5.7 NV_PVIC_THI_METHOD1

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_THI_METHOD1_DATA: 0: DATA_INIT (default) This contains method DATA which is to be sent to Falcon over method interface. Write to this register triggers write over Falcon method interface.

15.5.8 NV_PVIC_THI_INT_STATUS

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:1	R	0	Reserved
0	RW1C	0	NV_PVIC_THI_INT_STATUS_FALCON_INT: 0: FALCON_INT_INIT (default) 1: FALCON_INT_CLEAR Implies if there is any pending Falcon interrupt corresponding to an error condition.

15.5.9 NV_PVIC_THI_INT_MASK

Offset: 0x7c | Read/Write: R/W | Reset: 0x00000001

Bit	R/W	Reset	Description
31:1	R	0	Reserved
0	R/W	1	NV_PVIC_THI_INT_MASK_FALCON_INT: 1: FALCON_INT_INIT (default) When set, this bit enables generation of interrupts corresponding to Falcon error conditions

15.6 VIC Falcon Registers

15.6.1 HOSTIF Miscellaneous Registers

15.6.1.1 NV_PVIC_FALCON_ITFEN

Offset: 0x1048 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:3	R	0	Reserved
2	R/W	0	NV_PVIC_FALCON_ITFEN_PRIV_POSTWR. Indicates whether to use a post write on the main priv interface. By default, all writes to the Control and Status Bus (CSB) from the main priv interface are non-posted to support error reporting. 0: PRIV_POSTWR_INIT (default) 0: PRIV_POSTWR_FALSE 1: PRIV_POSTWR_TRUE
1	R/W	0	NV_PVIC_FALCON_ITFEN_MTHDEN. Method interface enable. When set, allows the host to push methods into the method FIFO 0: MTHDEN_INIT (default) 0: MTHDEN_DISABLE 1: MTHDEN_ENABLE
0	R/W	0	NV_PVIC_FALCON_ITFEN_CTXEN. Context switch interface enable. When set, allows the context switch state machine to react to incoming context switch requests from the host. 0: CTXEN_INIT (default) 0: CTXEN_DISABLE 1: CTXEN_ENABLE

15.6.2 Falcon UCTL Registers

15.6.2.1 NV_PVIC_FALCON_CPUCTL

Offset: 0x1100 | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:6	R	0	Reserved
5	R	Unknown	NV_PVIC_FALCON_CPUCTL_STOPPED: Indicates whether the CPU is currently in the stopped state. Falcon exits this state if a 1 is written to the STARTCPU bit or if an interrupt arrives on one of its 2 inputs and the corresponding IE bit in CSW is set. 1: STOPPED_TRUE 0: STOPPED_FALSE
4	R	Unknown	NV_PVIC_FALCON_CPUCTL_HALTED: Indicates whether the CPU is currently in the halted state. Falcon can only exit this state when a 1 is written to the STARTCPU bit. 1: HALTED_TRUE 0: HALTED_FALSE
3	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_HRESET: Set to TRUE to apply hard reset. This bit will auto-clear. Setting it FALSE has no effect. 1: HRESET_TRUE 0: HRESET_FALSE
2	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_SRESET: Set to TRUE to apply soft reset. This bit will auto- clear. Setting it FALSE has no effect. 1: SRESET_TRUE 0: SRESET_FALSE
1	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_STARTCPU: Set STARTCPU to TRUE to start CPU execution while in a HALTED state. If a start request is still pending, setting to FALSE will cancel the start request. Writing any value has no effect while the CPU is running. 1: STARTCPU_TRUE 0: STARTCPU_FALSE
0	RW1C	Unknown	NV_PVIC_FALCON_CPUCTL_IINVAL: Set to TRUE to mark all blocks in IMEM except block 0 as INVALID. This bit will auto-clear. Setting to FALSE has no effect. 1: IINVAL_TRUE 0: IINVAL_FALSE

15.6.2.2 NV_PVIC_FALCON_BOOTVEC

Offset: 0x1104 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_FALCON_BOOTVEC_VEC: Stores the initial execution start address of the CPU when it is first started after a reset. 0: VEC_INIT (default)

15.6.3 Falcon DMA Registers

15.6.3.1 NV_PVIC_FALCON_DMACTL

Offset: 0x110c | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:7	R	0	Reserved
6:3	R	None	NV_PVIC_FALCON_DMACTL_DMAQ_NUM: Indicates valid request number at DMA request queue.
2:1	R	0	Reserved
0	R/W	1	NV_PVIC_FALCON_DMACTL_REQUIRE_CTX: When set to TRUE, a valid context must be loaded before any DMA request can be serviced. Pending requests without a valid current context remain pending, and do not prevent the engine from reporting idle. When clear, DMA requests are serviced regardless of the current context. Once a request is issued, it must complete before the engine can report idle, as needed for example to process WFI context switch requests. 1: REQUIRE_CTX_INIT (default) 1: REQUIRE_CTX_TRUE 0: REQUIRE_CTX_FALSE

15.6.3.2 NV_PVIC_FALCON_DMATRFBASE

Offset: 0x1110 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_FALCON_DMATRFBASE_BASE: 0: BASE_INIT (default)

15.6.3.3 NV_PVIC_FALCON_DMATRFMOFFS

IMEM/DMEM offset for the transfer.

Offset: 0x1114 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:0	R/W	0	NV_PVIC_FALCON_DMATRFMOFFS_OFFS: 0: OFFS_INIT (default)

15.6.3.4 NV_PVIC_FALCON_DMATRFCMD

Offset: 0x1118 | Read/Write: R/W | Reset: 0x0000XXXX

Bit	R/W	Reset	Description
31:15	R	0	Reserved
14:12	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_CTXDMA:
11	R	0	Reserved

Bit	R/W	Reset	Description
10:8	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_SIZE: 0: SIZE_4B 1: SIZE_8B 2: SIZE_16B 3: SIZE_32B 4: SIZE_64B 5: SIZE_128B 6: SIZE_256B
7:6	R	0	Reserved
5	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_WRITE: 1: WRITE_TRUE 0: WRITE_FALSE
4	R/W	Unknown	NV_PVIC_FALCON_DMATRFCMD_IMEM: 1: IMEM_TRUE 0: IMEM_FALSE
3:2	R	0	Reserved
1	R	Unknown	NV_PVIC_FALCON_DMATRFCMD_IDLE: Indicates that the DMA engine is still busy with a transfer or has more transfers pending in the queue. 1: IDLE_TRUE 0: IDLE_FALSE
0	R	Unknown	NV_PVIC_FALCON_DMATRFCMD_FULL: Indicates that the DMA request queue is full and a valid request is still needed to move into the queue. 1: FULL_TRUE 0: FULL_FALSE

15.6.3.5 NV_PVIC_FALCON_DMATRFFBOFFS

Frame buffer (FB) offset for the transfer.

Offset: 0x111c │ Read/Write: R/W │ Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	NV_PVIC_FALCON_DMATRFFBOFFS_OFFS: 0: OFFS_INIT (default)

CHAPTER 16: CPU COMPLEX

The NVIDIA® Tegra® X1 processor CPU complex contains two multi-core CPU clusters. Each cluster contains four identical CPUs (four Cortex®-A57 CPUs in one and four Cortex-A53 CPUs in the other). All of the CPUs include the NEON Media Processing Engine. Each cluster has its own L2 cache which is shared with all cores of the same cluster.

Note: *The CPU subsystem supports a switch-cluster mode meaning that only one of the clusters can be active at any given time*

16.1 Cortex-A57 CPU Cluster

All four CPUs in Cluster 0 are identical Cortex-A57 implementations of ARM's v8 architecture. The CPU features include:

- Full implementation of ARMv8 architecture
- Superscalar, variable-length, out-of-order pipeline
- Dynamic branch prediction with Branch Target Buffer (BTB) and Global History Buffer RAMs, a return stack, and an indirect predictor
- 48-entry fully-associative L1 instruction TLB with native support for 4 KB, 64 KB, and 1 MB page sizes.
- 32-entry fully-associative L1 data TLB with native support for 4 KB, 4 KB and 1 MB pages sizes.
- 4-way set-associative unified 1024-entry Level 2 (L2) TLB in each processor
- Fixed 48 KB L1 instruction cache and 32 KB L2 data cache
- A 2 MB L2 cache shared by all the Cortex-A57 cores
- Embedded Trace Microcell (ETM) based on the ETMv4 architecture
- Performance Monitor Unit (PMU) based on the PMUv3 architecture
- Cross Trigger Interface (CTI) for multiprocessor debugging
- Cryptographic Engine for crypto function support
- Interface to an external Generic Interrupt Controller (vGIC-400)
- Support for power management with multiple power domains
- Cortex-A57 Revision r1p1--with the ECO fix for ARM bug # 829520 (current version at the time of writing)

Further information on the Cortex-A57 CPU is available in documentation ARM has published on their website. This information includes the Technical Reference Manual and Software Developers' Errata Notice.

16.2 Cortex-A53 CPU Cluster

All four CPUs in the second cluster are identical Cortex-A53 implementations of ARM's v8 architecture. The CPU features include:

- Full implementation of ARMv8 architecture instruction set
- In-order pipeline with dual-issue support for most instructions
- Fixed 32 KB L1 instruction cache and 32 KB L1 data cache
- A 512 KB L2 cache shared by all the Cortex-A53 cores. It provides coherency across all the Cortex-A53 cores.
- Embedded Trace Microcell (ETM) based on the ETMv4 architecture

- Performance Monitor Unit (PMU) based on the PMUv3 architecture
- Cross Trigger Interface (CTI) for multiprocessor debugging
- Cryptographic Engine for crypto function support
- Interface to an external Generic Interrupt Controller (vGIC-400)
- Support for power management with multiple power domains
- Cortex-A53 Revision: r0p2 (current version at the time of writing)

Further information on the Cortex-A53 CPU is available in documentation ARM has published on their website. This information includes the Technical Reference Manual and Software Developers' Errata Notice.

16.3 MSELECT Registers

16.3.1 MSELECT_CONFIG_0

Offset: 0x0 I Read/Write: R/W I Reset: 0x07ff4020 (0bx00001111111111110100000000100000)

Bit	Reset	Description
30	0x0	WRAP_TO_INCR_SLAVE3
29	0x0	WRAP_TO_INCR_SLAVE2 (GPU)
28	0x0	WRAP_TO_INCR_SLAVE1 (PCIe)
27	0x0	WRAP_TO_INCR_SLAVE0 (APC)
26	0x1	UNSUPP_SIZE_ERR_EN
25	0x1	ERR_RESP_EN_SLAVE2 (GPU)
24	0x1	ERR_RESP_EN_SLAVE1 (PCIe)
23	0x1	WRITE_TIMEOUT_EN_SLAVE3
22	0x1	READ_TIMEOUT_EN_SLAVE3
21	0x1	WRITE_TIMEOUT_EN_SLAVE2 (GPU)
20	0x1	READ_TIMEOUT_EN_SLAVE2 (GPU)
19	0x1	WRITE_TIMEOUT_EN_SLAVE1 (PCIe)
18	0x1	READ_TIMEOUT_EN_SLAVE1 (PCIe)
17	0x1	WRITE_TIMEOUT_EN_SLAVE0 (APC)
16	0x1	READ_TIMEOUT_EN_SLAVE0 (APC)
15	0x0	SAFE_MODE_MASTER1
14	0x1	ENABLE_GPU_APERTURE
13	0x0	ENABLE_APB_APERTURE
12:6	0x0	RESERVED2
5	0x1	ENABLE_PCIE_APERTURE
4:1	0x0	RESERVED1
0	0x0	SAFE_MODE_MASTER0

To allow WRAP burst type transactions for PCIe and GPU on the MMIO path:

Set bits 24:25 in MSELECT_CONFIG_0 register to 0bx00 in order to disable unsupported type check.

Set bits 29:28 in MSELECT_CONFIG_0 register to 0bx11 in order to enable WRAP type and its conversion to INCR type.

16.3.2 MSELECT_ERROR_STATUS_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx00000000000)

Bit	Reset	Description
10	CLEAR	UNSUPP_SIZE_ERR: i=1 0 = CLEAR 1 = SET
9	CLEAR	ERR_SLAVE2: i=1 0 = CLEAR 1 = SET

Bit	Reset	Description
8	CLEAR	ERR_SLAVE1: i=1 0 = CLEAR 1 = SET
7	CLEAR	WRITE_TIMEOUT_SLAVE3: i=0 0 = CLEAR 1 = SET
6	CLEAR	READ_TIMEOUT_SLAVE3: i=0 0 = CLEAR 1 = SET
5	CLEAR	WRITE_TIMEOUT_SLAVE2: i=0 0 = CLEAR 1 = SET
4	CLEAR	READ_TIMEOUT_SLAVE2: i=0 0 = CLEAR 1 = SET
3	CLEAR	WRITE_TIMEOUT_SLAVE1: i=0 0 = CLEAR 1 = SET
2	CLEAR	READ_TIMEOUT_SLAVE1: i=0 0 = CLEAR 1 = SET
1	CLEAR	WRITE_TIMEOUT_SLAVE0: i=0 0 = CLEAR 1 = SET
0	CLEAR	READ_TIMEOUT_SLAVE0: i=0 0 = CLEAR 1 = SET

CHAPTER 17: FLOW CONTROLLER

The flow controller provides the sequencing of hardware-controlled CPU power states for the main CPU complex and the ARM7 coprocessor (also known as BPMP-Lite, and historically AVP or COP).

17.1 Flow Controller Sequences

The flow controller orchestrates several complex sequences. The following subsections describe those sequences in detail.

17.1.1 CPU Power Gating/Ungating Sequence

The Cortex-A57 processor's power-gating can be triggered by either i) a direct register write to PMC or ii) interacting with the flow controller. This section describes the power-gating sequence via flow-controller interaction.

The flow controller interfaces with the Processors, PMC, and CAR to implement power-gating on/off hardware-sequence. The power-gating controller is in the PMC, while clock and reset controls are in the CAR.

17.1.1.1 CPU Power-Gating (Power OFF) Sequence

- [SW-CPU] Prepare processor (flush CPU L1$, context save, set up flow controller) for power-gating
- [SW-CPU] Execute WFI to make sure there is no outstanding processor transaction
- [HW-CPU] Assert *STANDBYWFI* to the flow controller
- [HW-Flow] If INTERCEPT_ENTRY_PG_COREx_ENABLE is set, the flow controller generates an interrupt to LIC. The interrupt is routed to ARM7/BPMP-Lite. The flow controller waits for FC_SEQUENCE_INTERCEPT.INTERRUPT_PENDING_COREx to be cleared.
- [HW-Flow] Assert flow2car_*_rst
  - This is not used in power gate sequence when powergate_enable is high
  - When power-gate enable is "0", the CAR uses this to assert CPU reset
- [HW-Flow] Request the PMC (by asserting flow2pmc_req and associated bus) to power-off the CPU
- [HW-PMC] Assert the clamp control signal. This goes to ccplex and also goes to the CAR.
- [HW-CAR] Assert CPU reset and reset the CPU clock
  - [HW-CPU] Stop CPU clock
- [HW-CAR] Ack PMC to indicate CPU clock is stopped
- [HW-PMC] Power off CPU (assert CPU PG-enables)
- [HW-PMC] Clear PWRGATE_STS register bit. This is to indicate that CPU is power-gated.
- [HW-PMC] Ack to the flow controller to confirm power-off (by asserting pmc2flow_ack)

Note: *After receiving ack from the PMC, the flow controller must not issue a new request to the PMC until pmc2flow_ack goes idle.*

- [HW-Flow] Deassert flow2pmc_req (after seeing the Ack)
- [HW-PMC] Deassert Ack (after seeing request deassertion)
- [HW-Flow] Do not start another request until Ack goes down

17.1.1.2 CPU Power-Ungating (Power ON) Sequence

The generic CPU power-ungating (power OFF) sequence is described below. The specific details of main CPU (Cortex-A15) are described in their respective power-gating sections.

- [HW-Flow] After detecting wake up condition
- [HW-Flow] If INTERCEPT_EXIT_PG_COREx_ENABLE is set, The flow controller generates an interrupt to LIC. The interrupt is routed to ARM7/BPMP-Lite. The flow controller waits on FC_SEQUENCE_INTERCEPT.INTERRUPT_PENDING_COREx to be cleared by the BPMP-Lite through a register write.
  - This sequence is applicable to the main CPU (Cortex-A57 or Cortex-A9) core only.
- [HW-Flow] Request PMC to power-on CPU (by asserting flow2pmc_req and associated bus)
- [HW-PMC] Power-on the CPU (deassert CPU PG-enables)
- [HW-PMC] Set PWRGATE_STS register bit. This indicates that the CPU is power-ungated
- [HW-PMC] Ack flow controller (by asserting pmc2flow_ack) that CPU is powered on
- [HW-Flow] Deassert flow2car_*_rst to CAR
- [HW-CAR] Request the CPU to ungate the CPU clock (by deasserting car2ce*_cke)
- [HW-CPU] Ungate the CPU clock
- [HW-CAR] Request the PMC to remove CPU clamp (by asserting car2pmc_ack)
  - This makes sure that the clock gets ungated well ahead of reset deassertion
- [HW-PMC] Removes the clamping (by deasserting pmc2ce*_clamp)
  - This goes to the CPU and also to the CAR
- [HW-PMC] Ack the flow controller (by deasserting pmc2flow_ack)
- [HW-CAR] Deasserts reset
- [SW-CPU] CPU fetches code from reset vector.

Note: *Note all of the Cortex-A57 CPUs share one reset-vector register (in the EVP registers).*

17.1.2 Non-CPU Power Gating/Ungating Sequence

From a software perspective, non-CPU power gating/ungating is done as part of CPU power gating/ungating request. The flow controller issues a separate request (to the PMC) for power-gating CPU and non-CPU domains. The main reason is that the PMC can only sequence one power domain at a time, so the flow controller serializes requests on behalf of the PMC. From the flow controller hardware perspective, the sequence for non-CPU domain power gating is similar to CPU domain power gating/ ungating which is described in above sections.

As part of the non-CPU power gating sequence, prior to power gating the non-CPU logic (L2, etc.), the flow controller will need to flush/invalidate the L2 cache through the L2FLUSHREQ/L2FLUSHDONE interface.

17.1.3 CPU Rail Gating/Ungating Sequence

This section describes CPU rail gating/ungating with the help of the flow controller.

From a software perspective, the CPU rail is powered on/off as part of CPU power-gating on/off request. The flow controller issues a separate request (to the PMC) for CPU power gating/ungating and CPU rail gating/ungating. The main reason is that the PMC can only sequence one power-domain at a time, so the flow controller serializes requests on behalf of the PMC. From flow controller hardware perspective, the sequence for CPU rail gating is similar to individual CPU power gating, which is described in the above sections. However, as part of CPU rail-ungating, the flow controller also triggers RAM re-repair as described below.

17.1.3.1 CPU Rail Ungating Steps

- [HW-Flow] Request the PMC to power-on the CPU rail
- [HW-Flow] Wait for PMC acknowledgment
- [HW-Flow] Request the Re-shift block (also known as the re-repair block) to re-repair all of the segments of CPU repair chain
  - [HW-Reshift] Performs repair of all segments of CPU chain in parallel
- [HW-Flow] Wait for re-repair acknowledgments (from all segments)
- [HW-CAR] Deassert CPU cluster0 reset

17.1.3.2 Cluster RAM Re-Repair

At (cold or warm) boot power-up of the CPU rail, software powers up the CPU rail (by direct register writes to the PMC or the PMIC), and triggers RAM repair by writing to the RAM_REPAIR[REQ] register of the flow controller. After that software polls RAM_REPAIR[STS] to ensure that the repair is done.

After boot power-up of the CPU rail, whenever the CPU rail is powered up by the flow controller, the flow-controller hardware also requests cluster RAM re-repair, and ensures that the repair is completed.

The re-repair request and ack signals are per segment of the CPU repair chain. The request, once asserted (high), must remain high for as long as it does not receive the corresponding ack signal.

17.1.4 CC4 Power State Sequence

The following is a high-level description of the CC3 HVC and Retention entry and exit sequences. A few important items to highlight:

- Entry and Exit from CC3 HVC is entirely driven by fixed-function hardware.
- Entry and Exit from CC4 Retention is aided by BPMP-Lite processor.
- The cluster always enters CC3 HVC state prior to entering Retention state.

Entry Sequence

- [SW-CPU] Software executes WFI on a given core to bring the core into idle state (C1 or C6)
  - Once the core asserts STANDBYWFI for the specific core, a hardware-based counter (in the flow controller) is initialized to CC4_CORE<x>_CTRL.CORE_IDLE_TIMER and starts counting down.
- [HW-CPU] The last core standing asserts STANDBYWFI indicating that it is idle and will enter an idle power state (C1 or C6).
  - As with all the other cores, once the core asserts STANDBYWFI for the specific core, a hardware-based counter (in the flow controller) is initialized to CC4_CORE<x>_CTRL.CORE_IDLE_TIMER and starts counting down.
  - If PG (C6) is requested, the core is allowed to enter and complete power-gating sequence.
- For the cores that are allowed to participate in the decision on whether to enter HVC (i.e., CC4_CORE<x>_CTRL.TIMER_COUNTDOWN_VALID is set), if any CC4_CORE<x>_CTRL.CORE_HVC_ENABLE is not set, then the cluster does not enter HVC and remains in CC1.
- If the min(CC4_CORE<x>_CTRL.CORE_IDLE_TIMER) is greater than CC3_HVC_CONTROL. HVC_RES_TIME_THRESHOLD, then the cluster is allowed to enter CC3 HVC.
- [HW-Flow] The flow controller checks the status of the CPUQACTIVE signal, one per core. If CPUQACTIVE is de-asserted, the flow controller asserts the CPUQREQ signal requesting a transition to the “retention” state for each core. The flow controller does the same with the L2 cache.
  - Note that ARM refers to the CPUQx interface as “retention” interface

- Note that the flow controller is required to handshake only with the cores that are in C1. No handshake is required (or possible) for cores that are in WFI and power gated (=C6). The flow controller is fully aware which cores have been power-gated up to this point.
- [HW-CPU] If cores and L2 cache are still idle, they accepts the request and assert CPUQACCEPTn and L2QACCEPTn signal, otherwise they assert the CPUQDENYn and L2QDENYn signals, respectively.
- [HW-Flow] If denied, the flow controller de-asserts CPUQREQ and L2QREQ signals and retries the previous two steps again after the programmable time period
  - CC3_HVC_RETRY. THRESHOLD represents the retry interval. The down-counting counter initialized to this threshold runs on SCLK
- [HW-Flow] If CPUQACCEPTn and L2QACCEPTn are asserted, the flow controller is free to proceed with the CC4 entry sequence.
- [HW-Flow] If FC_SEQUENCE_INTERCEPT. INTERCEPT_HVC_ENABLE set, generate an interrupt and wait on FC_SEQUENCE_INTERCEPT. HVC_INTERRUPT_PENDING to be cleared.
  - BPMP can potentially clock gate the secondary clocks coming into VDD_CPU so that they do not become a limiting factor to Vmin.
- [HW-Flow] The flow controller asserts the Skipper3 override signal
  - This is a request to SoC Therm to initiate pulse skipping to scale down the CPU frequency.
  - SoC Therm uses pre-configured skipper control values
  - flow2soctherm_skipper3_override is kept asserted
- [HW-Flow] The flow controller triggers the CL_DVFS logic via a dedicated interface. It asserts the flow2cldfvs_override signal.
  - CL_DVFS transitions to open-loop mode. Voltage control is disabled.
- [HW-Flow] The flow controller issues a request to the PMC to lower the voltage
  - This represents a new CPU_ID encoding on the FC to PMC (VDD_CPU) rail control interface
- [HW-PMC] The PMC disables the TSOC polling/sensing on VDD_CPU rail
- [HW-PMC] The PMC requests VDD_LOW from PMIC via CPU_PWR_REQ interface
  - This assumes that VDD_LOW in the PMIC has properly been set to Vmin by software prior to the start of the HVC sequence
- [HW-PMC] The PMC notifies the flow controller that the rail voltage change request has been issued
- [HW-FC] The flow controller enters CC4 HVC state
  - The flow controller sets CC3_HVC_CONTROL.CC3_HVC_ENTERED to indicate that HVC state achieved
  - The flow controller monitors the CPUQACTIVE and L2QACTIVE signals for activity
- [HW-FC] Check to see if (INTERCEPT_ENTRY_CC4_ENABLED)
  - If INTERCEPT_ENTRY_CC4_ENABLED is not set, the flow controller does not attempt to enter the Retention state and continues to monitor CPUQACTIVE and L2QACTIVE signals for activity
- For the cores that are allowed to participate in the decision on whether to enter Retention (i.e., CC4_CORE<x>_CTRL.TIMER_COUNTDOWN_VALID is set), if any CC4_CORE<x>_CTRL.CORE_RET_ENABLE is not set, then the cluster does not enter the Retention state and remains in HVC.
- If the min(CC4_CORE<x>_CTRL.CORE_IDLE_TIMER) is greater than CC4_RET_CONTROL. RET_RES_TIME_THRESHOLD, then the cluster is allowed to enter CC4 Retention.
- [HW-FC] If the above conditions are met, then the flow controller moves to the next step. Otherwise, the flow controller enters a standby state monitoring the CPU's xQACTIVE signals:
- [HW-FC] The flow controller issues an interrupt to LIC

  - LIC routes the interrupt to ARM7 (BPMP-Lite)
- At this point, ARM7/BPMP can attempt to bring the CPU into CC4 Retention (VDD_CPU = Vret)
- BPMP-Lite takes control of the sequence and determines in the background whether to enter CC4_RETENTION
- [BPMP-Lite] If a pending interrupt in FC (CC4_FC_STATUS.INT_STATUS), BPMP-Lite aborts the sequence to enter CC4 Retention
- [BPMP-Lite] Disable the LIC-GIC interface by writing to the LIC state register. Disable legacy IRQ and FIQ in the flow controller
- [BPMP-Lite] Instruct the flow controller to not process further interrupts
- Set CC4_FC_STATUS.CC4_WAKE_MASK register
- [BPMP-Lite] Wait for a fixed delay. This delay will make sure that any in flight interrupt is seen by the flow controller.
- [BPMP-Lite] Check the pending interrupt in the flow controller (CC4_FC_STATUS.INT_STATUS), BPMP-Lite aborts the sequence to enter CC4 Retention
- [BPMP-Lite] Enables CPU clamp signals. Disables clock to the CPU.
- [BPMP-Lite] Writes 0x0 to the CC3_HVC_DFLL_CTRL_MODE register in the CL_DVFS. The flow controller keeps the flow2cldvfs_override signal asserted. This write disables CL_DVFS (as long as override signal is kept asserted).
- [BPMP-Lite] Sets VDD_LOW in the PMIC to Vret
- [BPMP-Lite] Clears the interrupt in the flow controller by writing to CC3_HVC_CONTROL.INTERRUPT_PENDING.
  - This step makes sure that BPMP stays uninterrupted through the retention entry sequence.

Exit Sequence

- [HW-CPU] Assertion of CPUQACTIVE (or L2QACTIVE) is an indication that the CPU is requesting exit from the HVC state. Note that if the Retention state has been entered, the CPU and L2 are inactive (= no running clock). The flow controller will need to also monitor wake events from LIC or SoC.
- [HW-Flow] If CC4_WAKE_MASK is set, we are coming out of the Retention state (not CC4 Vmin). If CC4_FC_STATUS.CC4_WAKE_MASK is set, the flow controller issues an interrupt.
- [HW-Flow] Waits for CC4_FC_STATUS.CC4_WAKE_MASK to be cleared by BPMP-Lite
- [BPMP-Lite] Updates the VDD_LOW (in the PMIC) to Vmin.
- [BPMP-Lite] Writes 0x1 to the FC_DFLL_CTRL_MODE register. The flow controller keeps the override signal to CLDVFS asserted. Enables CL_DVFS and puts it into open loop mode.
- [BPMP-Lite] Waits for the timer to allow for voltage ramp up. Turns on CCPLEX clocks. De-asserts the clamps. Enables the LIC->GIC interface. Clears the CC4_FC_STATUS.CC4_WAKE_MASK register in the flow controller.
- [HW-Flow] De-asserts the SoC Therm override control signal
- [HW-Flow] The flow controller requests CPU rail power up from the PMC
- [HW-PMC] Asserts CPU_PWR_REQ from the PMIC. Wait on timer to ensure that the rail is up.
- [HW-PMC] Enable TSOSC
- [HW-PMC] Send Ack back to the flow controller as a response to the earlier CPU rail up request.
- [HW-Flow] If coming out of CC4 HVC only, if FC_SEQUENCE_INTERCEPT. INTERCEPT_HVC_ENABLE set, generate an interrupt and wait on FC_SEQUENCE_INTERCEPT. HVC_INTERRUPT_PENDING to be cleared.
  - The BPMP will ungate the secondary clocks into VDD_CPU, if they were gated in the first place at entry into the CC3_HVC sequence.
- [HW-FC] De-asserts the override signal to CL_DVFS. The CL_DVFS goes back to closed-loop mode. Voltage control is enabled.

17.2 Flow Controller Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

In the Flow Controller register descriptions, CPU refers to CPU0, and BPMP-Lite refers to the ARM7 coprocessor.

The EVENTS register is replicated per core, once for BPMP-Lite, four times for the CPU cores.

The fields and enums have the following interpretation:

- MODE defines how the flow controller logic operates; the reset value is 0x0 for CPU0 and BPMP-Lite (does nothing) and 0x2 for CPU1, 2, 3 (WAITEVENT). This is related to the default behavior of the reset generator.
- Enumerated values for the field MODE are defined below.

Field Mode	Enumerated Value	Description
FLOW_MODE_NONE	0	No flow control
FLOW_MODE_RUN_AND_INT	1	Keep running but generate interrupt when event conditions met (not used)
FLOW_MODE_WAITEVENT	2	Stop running until event conditions met
FLOW_MODE_WAITEVENT_AND_INT	3	Same as FLOW_MODE_WAITEVENT but generate an interrupt when resumed (not used)
FLOW_MODE_STOP_UNTIL_IRQ	4	Stop until an interrupt controller interrupt occurs
FLOW_MODE_STOP_UNTIL_IRQ_AND_INT	5	Same as FLOW_MODE_STOP_UNTIL_IRQ but generate another interrupt when resumed (not used)
FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ	6	Stop until event conditions met AND an interrupt controller interrupt occurs

---

Note: *There is no enum for 7.*

The rest of the fields define which conditions will be taken into account to restart a halted processor.

There are two types of events:

- Counted events: Decrement the field called ZERO. Flow controller resumes when this counter reaches 0
- Activity events: The flow controller resumes when the activity occurs, no counting

17.2.1 FLOW_CTLR_HALT_CPU0/1/2/3_EVENTS_0

HALT_CPUx_EVENTS_0 offsets are as follows:

FLOW_CTLR_HALT_CPU0_EVENTS_0: Offset: 0x0

FLOW_CTLR_HALT_CPU1_EVENTS_0: Offset: 0x14

FLOW_CTLR_HALT_CPU2_EVENTS_0: Offset: 0x1c

FLOW_CTLR_HALT_CPU3_EVENTS_0: Offset: 0x24

Each of the four blocks has the same registers and bits.

Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ Bit 31: Halt = Halt until an event (e.g., interrupt). This bit is set by software and cleared by hardware when wake event is received by hardware. Bit 30: Wait = Wait until event counter counts down to zero. This bit is set by software and cleared by hardware when wait event is observed by hardware. Bit 29: Sync = This bit causes the NEXT Flow Controller reg read to stall the processor until the wake event occurs (Obsolete for CPUs. May be used for COP).
28	0x0	JTAG: Resume on JTAG activity
27	0x0	SCLK: Resume on Nth SYSCLK cycle ticks
26	0x0	X32K: Resume on Nth X32K clock input ticks
25	0x0	uSEC: Resume on Nth µs clock ticks
24	0x0	mSEC: Resume on Nth ms clock ticks
23	0x0	SEC: Resume on Nth second RTC clock ticks
22	0x0	X_RDY: Resume on Nth XIO.RDY Ext. IO Ready events
21:20	0x0	SMP31/0: Resume on Nth SMP.3[1,0] Semaphore set events
19:16	0x0	XRQ_D/C/B/A: Resume on Nth XRQ.[D,C,B,A] External Trigger events
15:12	0x0	OBE/F,IBE/F: Resume on Nth [O,I]B[E,F] [Outbox, Inbox] [Empty, Full] Events
11	0x0	LIC_IRQn: Resume on Legacy Interrupt Controller IRQ interrupt For CPU0, wake on cpu_nirq0 For CPU1, wake on cpu_nirq1 For CPU2, wake on cpu_nirq2 For CPU3, wake on cpu_nirq3 For CPU4, wake on the cpu_nirq which corresponds to CPU4 CPU-ID.
10	0x0	LIC_FIQn: Resume on Legacy Interrupt Controller FIQ interrupt. See LIC_IRQn for the description of <n>.
9	0x0	GIC_IRQn: Resume on MPCore GIC IRQ interrupt For CPU0, wake on MPCore nirq0 For CPU1, wake on MPCore nirq1 For CPU2, wake on MPCore nirq2 For CPU3, wake on MPCore nirq3 For CPU4, wake on MPCore_LP nirq.
8	0x0	GIC_FIQn: Resume on MPCore GIC FIQ interrupt. See GIC_IRQn for the description of <n>.
7:0	0x0	ZERO: Initialized then decremented. Note: If more than one event is enabled, the event counter will decrement based on an OR condition of enabled events.

17.2.2 FLOW_CTLR_HALT_COP_EVENTS_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K

Bit	Reset	Description
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

17.2.3 FLOW_CTLR_CPU0/1/2/3_CSR_0

The CPU_CSR registers are replicated for each CPU cores. They define additional characteristics of the flow controller linked to CPU cores, especially the interaction of the flow controller with power management functions. The LP1 state is normally entered and exited via side effects of the flow controller operation.

All fields of the CPU_CSR register have an initial value of 0.

FLOW_CTLR_CPUx_CSR_0 offsets are as follows:

FLOW_CTLR_CPU0_CSR_0: Offset: 0x8

FLOW_CTLR_CPU1_CSR_0: Offset: 0x18

FLOW_CTLR_CPU2_CSR_0: Offset: 0x20

FLOW_CTLR_CPU3_CSR_0: Offset: 0x28

Each of the four blocks has the same registers and bits.

Read/Write: R/W │ Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE: Current state of the PowerGate State Machine
23	RO	X	WAIT_EVENT: CPU is waiting, wake up is via an event
22	RO	X	HALT: CPU is halted – This bit is asserted when STANDBYWFI is asserted.
21	RO	X	P2F_ACK: pmc2flow_ack signal, this is the same signal for all CPUs
20	RO	X	F2P_PWRUP: flow2pmc_pwrup, this is the same signal for all CPUs
19	RO	X	F2P_REQ: flow2pmc_req valid, this is the same signal for all CPUs
17	RO	X	F2C_MPCORE_RST: TRUE when Requesting Reset of MPCore
15	RW	0x0	INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear
14	RW	0x0	EVENT_FLAG: TRUE when Event is Active -- Write-1-to Clear

Bit	R/W	Reset	Description
13:12	RW	0x0	ENABLE_EXT: If ENABLE is TRUE, then this specifies what to power off. 00b: PowerGate CPU only. 01b: PowerGate CPU and non-CPU (note, this option must be used only when last CPU is being power-gated) 10b: PowerGate CPU and turn CPU rail off (note, this option is valid for cluster0 CPU, and must be used only when last CPU is being power-gated). 11b: PG Emulation. Note, this option is for CPU-only PG emulation except for cluster-switch case (i.e., with SWITCH_CLUSTER=1) in which emulation applies to all PG steps of source cluster. In PG emulation, PG partition is reset but not power-gated.
11:8	RW	0x0	WAIT_WFI_BITMAP: All cores indicated in bitmap must be in STANDBY_WFI before CPU PowerGating.
7:4	RW	0x0	WAIT_WFE_BITMAP: All cores indicated in bitmap must be in STANDBY_WFE before CPU PowerGating. This is deprecated and shouldn't be used.
3	RW	0x0	IMMEDIATE_WAKE: If set, CPU is powered up immediately (without waiting for int or event)
2	RW	0x0	SWITCH_CLUSTER: If set, the active cluster will be switched when all indicated CPU reach STANDBY_WFI. This bit self clears once the cluster switch sequence is completed.
1	RW	0x0	EVENT_ENABLE: Generates an event when the flow controller exits the halted state
0	RW	0x0	ENABLE: PowerGate Enable. STANDBYWFx causes power-gating based on ENABLE_EXT values. Note 1: Software sets this bit to '1' for requesting CPU power-gating. CPU is power-gated based on the value of ENABLE_EXT field. Note 2: This bit is set by software and cleared by hardware (when power-gating sequence is completed).

17.2.4 FLOW_CTLR_COP_CSR_0

BPMP-Lite Control/Status for Interrupts

R/W addr=6000:700c

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx)

Bit	Reset	Description
15	0x0	INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear

17.2.5 FLOW_CTLR_XRQ_EVENTS_0

These events are not used as flow controller wake events.

XRQ Event Detect Selector Register

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	XRQ_D7_XRQ_D0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port D. The assertion level is determined by GPIO_INT.LVL.D. If more than one XRQ.D bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.D bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
23:16	0x0	XRQ_C7_XRQ_C0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port C. The assertion level is determined by GPIO_INT.LVL.C. If more than one XRQ.C bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.C bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
15:8	0x0	XRQ_B7_XRQ_B0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port B. The assertion level is determined by GPIO_INT.LVL.B. If more than one XRQ.B bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.B bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
7:0	0x0	XRQ_A7_XRQ_A0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port A. The assertion level is determined by GPIO_INT.LVL.A. If more than one XRQ.A bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.A bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.

17.2.6 FLOW_CTLR_CLUSTER_CONTROL_0

This register is reserved.

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x04004000 (0b00000100000000000001000000xxxxxxx0)

Bit	Reset	Description
31:20	0x40	POST_SWITCH_DELAY
19:8	0x40	PRE_SWITCH_DELAY
0	0x0	ACTIVE

17.2.7 FLOW_CTLR_CPU_PWR_CSR_0

Offset: 0x38 │ Read/Write: R/W │ Reset: 0x000001XX (0bxxxxxxxxxxxxxxxxx00000001xxxx0xx0)

Bit	R/W	Reset	Description
15	RW	0x0	DBG_STS_EN:
14:13	RW	0x0	DBG_C1NC_STS:
12:11	RW	0x0	DBG_C0NC_STS:
10:9	RW	0x0	DBG_RAIL_STS:
8	RW	0x1	CPU_RG_CFG: If this bit is set, then the flow controller would initiate last-CPU PG (along with CPU RG (rail-gating) or non-CPU PG) only when other 3 CPUs are already power-gated. This is primarily for debug purposes. 0 = DISABLE 1 = ENABLE
7:6	RO	X	C1NC_STS: Hardware updates this register based on the current status of the C1NC partition. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note that the flow controller requests C1NC power on/off based on this register value or PMC power gate-status of C1NC depending upon the USE_FLOW_STS register. 0 = PARTITION_OFF 1 = PG_IN_PROGRESS 2 = PU_IN_PROGRESS 3 = PARTITION_ON
5:4	RO	X	C0NC_STS: hardware updates this register based on the current status of the C0NC partition. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note, flow controller requests C0NC power on/off based on this register value or PMC pwrgate-status of C0NC depending upon USE_FLOW_STS register. 0 = PARTITION_OFF 1 = PG_IN_PROGRESS 2 = PU_IN_PROGRESS 3 = PARTITION_ON
3	RW	0x0	USE_FLOW_STS:If this is set (=1), then the flow controller will use C0NC_STS/ C1NC_STS/ RAIL_STS bits to determine whether to power-gate C0NC/C1NC/CRAIL domains. If this bit is clear (=0), then the flow controller will use the PMC powergate-status of C0NC/C1NC/CRAIL to determine whether to power-gate the C0NC (and C1NC) domains. In either case, the C0NC_STS/C1NC_STS/RAIL_STS fields are updated in the same way.
2:1	RO	X	RAIL_STS: Hardware updates this register based on the current status of the CPU-rail. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note, flow controller requests CPU-RAIL power on/off based on this register value or PMC power gate-status of CRAIL depending upon USE_FLOW_STS register. 0 = RAIL_OFF 1 = RG_IN_PROGRESS 2 = RU_IN_PROGRESS 3 = RAIL_ON
0	RW	0x0	RAIL_ENABLE: CPU rail power-on request. Software sets this to request CPU rail power-on by flow-controller interaction. If rail is already on, writing to this is ignored. Hardware clears this bit when CPU power rail in turned on.

17.2.8 FLOW_CTLR_MPID_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	CPU_ID: CPU-ID of CPULP. Software programs this field before cluster0 -> cluster1 switch. This ID is provided to MPCore_LP which is what is read when the OS reads MPIDR. Also, the flow controller uses this to identify wake interrupts to be used for CPULP wakeup.

17.2.9 FLOW_CTLR_RAM_REPAIR_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxx00000000xxxx01x0)

Bit	R/W	Reset	Description
23:16	RO	X	DBG_STS: Repair done (acknowledge) from repair logic.
15:8	RW	0x0	DBG_REQ: Repair request for individual segments.
3	RW	0x0	DBG_EN: Debug enable to be able to repair of individual segments of the cluster0 repair chain 0 = DISABLE 1 = ENABLE
2	RW	0x1	BYPASS_EN: RAM repair bypass enable. 0: Flow-controller requests RAM-repair at CPU RAIL power-up. 1: Flow-controller does not request RAM-repair at CPU RAIL power-up 0 = DISABLE 1 = ENABLE
1	RO	X	STS: Indicates Cluster repair chain status. hardware sets this to '1' when repair of all segments is done. hardware clears this to '0' when RAM repair is requested.
0	RW	0x0	REQ: Cluster0 RAM repair request. Software sets this bit to '1' to initiate RAM repair request to all segments in parallel. Hardware clears this bit when software triggered repair is done (i.e., STS=1) 0 = DISABLE 1 = ENABLE

17.2.10 FLOW_CTLR_FLOW_DBG_SEL_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	R/W	Reset	Description
19:18	RO	X	RG_PWR_STATE: Current state of Rail Gate state machine
17:16	RO	X	RU_PWR_STATE: Current state of Rail Ungate state machine
15:14	RO	X	NC_PG_PWR_STATE: Current state of Non CPU Power Gate state machine
13:12	RO	X	NC_PU_PWR_STATE: Current state of Non CPU Power Ungate state machine

Bit	R/W	Reset	Description
11:8	RW	0x0	CNT1_SEL: Activity selection which would be counted by CNT1. **Note** power/rail activities are counted only when power/rail gating is via flow controller (and not direct PMC register write). 0: Idle (counter holds its current value) 1: Increment by 1 every time CPU0 is power gated 2: Increment by 1 every time CPU1 is power gated 3: Increment by 1 every time CPU2 is power gated 4: Increment by 1 every time CPU3 is power gated 5: Increment by 1 every time and of CPU0/1/2 or 3 is power-gated 6: Increment by 1 every time CPU-LP is power gated 7: Increment by 1 every time Ccluster0 non-CPU is power gated 8: Increment by 1 every time Cluster1 non-CPU is power gated 9: Increment by 1 every time CPU RAIL is powered off 10: Increment by 1 every time cluster0 is switched to cluster1 11: Increment by 1 every time cluster1 is switched to cluster0 12: Increment by 1 every time HVC is entered 13: Increment by 1 every time Retention is entered 0 = IDLE 1 = CPU0_PG 2 = CPU1_PG 3 = CPU2_PG 4 = CPU3_PG 5 = CPU0123_PG 6 = CPULP_PG 7 = C0NC_PG 8 = C1NC_PG 9 = CRAIL_OFF 10 = C0_TO_C1_SWITCH 11 = C1_TO_C0_SWITCH 12 = HVC_ENTRY 13 = RETENTION_ENTRY
3:0	RW	0x0	CNT0_SEL: Activity selection which would be counted by CNT0. **Note:** Power/rail activities are counted only when power/rail gating is via the flow controller (and not direct PMC register write). 0: Idle (counter holds its current value) 1: Increment by 1 every time CPU0 is power gated 2: Increment by 1 every time CPU1 is power gated 3: Increment by 1 every time CPU2 is power gated 4: Increment by 1 every time CPU3 is power gated 5: Increment by 1 every time and of CPU0/1/2 or 3 is power-gated 6: Increment by 1 every time CPU-LP is power gated 7: Increment by 1 every time Ccluster0 non-CPU is power gated 8: Increment by 1 every time Cluster1 non-CPU is power gated 9: Increment by 1 every time CPU RAIL is powered off 10: Increment by 1 every time cluster0 is switched to cluster1 11: Increment by 1 every time cluster1 is switched to cluster0 12: Increment by 1 every time HVC is entered 13: Increment by 1 every time retention is entered 0 = IDLE 1 = CPU0_PG 2 = CPU1_PG 3 = CPU2_PG 4 = CPU3_PG 5 = CPU0123_PG 6 = CPULP_PG 7 = C0NC_PG 8 = C1NC_PG 9 = CRAIL_OFF 10 = C0_TO_C1_SWITCH 11 = C1_TO_C0_SWITCH 12 = HVC_ENTRY 13 = RETENTION_ENTRY

17.2.11 FLOW_CTLR_FLOW_DBG_CNT0_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: Activity count value (updated by hardware based on CNT0_SEL). This register can be written by software to be able to clear it.

17.2.12 FLOW_CTLR_FLOW_DBG_CNT1_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: Activity count value (updated by hardware based on CNT1_SEL). This register can be written by software to be able to clear it.

17.2.13 FLOW_CTLR_FLOW_DBG_QUAL_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0bxx00xx00xxx00000xxx00000xxx00000)

Bit	Reset	Description
29	0x0	IRQ2CCPLEX_ENABLE:Qualifier for legacy IRQ 0 = DISABLE 1 = ENABLE
28	0x0	FIQ2CCPLEX_ENABLE:Qualifier for legacy FIQ 0 = DISABLE 1 = ENABLE
25:24	0x0	AXICIF_CG_DIS: CCPLEX AXICIF/MCCIF clock gating disable. Bit 0: Fast cluster rclk-gating disable (1=DISABLE, 0=ENABLE). Bit 1: Fast cluster wclk-gating disable (1=DISABLE, 0=ENABLE).
20:16	0x0	PWRUPREQ_QUAL: CPU DBGPWRUPREQ qualifier. Bit encoding similar to PWRDNREQ_QUAL.
12:8	0x0	NOPWRDWN_QUAL: CPU DBGNOPPWRDN qualifier. Bit encoding similar to PWRDNREQ_QUAL. Note: Multi cycle paths are defined on flow2car*_pg_enable andflow2car_mpcore*_rst. pg_enable should be stable before flow2car_mpcore*_rst gets asserted. For the "PG Emulation" case, this delay should be handled in software. ccplex2flow_dbgnopwrdwn also acts as pg_emulation case. This should be programmed before.
4:0	0x0	PWRDNREQ_QUAL: CPU DBGPWRDNREQ qualifier. Bit 0 for CPU0. Bit 1 for CPU1. Bit 2 for CPU2. Bit 3 for CPU3. Bit 4 for CPU-LP (also known as shadow CPU).

17.2.14 FLOW_CTLR_FLOW_CTLR_SPARE_0

Offset: 0x54 | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:16	0xffff	SPARE_HI
15:0	0x0	SPARE_LO

17.2.15 FLOW_CTLR_FC_SEQUENCE_INTERCEPT_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
21	0x0	INTERCEPT_HVC_ENABLE: If set, generates an interrupt to LIC before starting the HVC sequence and after exiting the HVC sequence. This interrupt is routed to ARM7/BPMP-Lite. 0 = DISABLE 1 = ENABLE
20	0x0	INTERCEPT_ENTRY_CC4_ENABLE: If set, generates an interrupt to LIC after the HVC state has been entered. This interrupt is routed to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	INTERCEPT_ENTRY_PG_NONCPU_ENABLE: 0: State machine does not halt at the start of the non-CPU power gating sequence 1: State machine halts at the start of the non-CPU power gating sequence. Issues an interrupt to ARM7/BPMP-Lite. 0 = DISABLE 1 = ENABLE
18	0x0	INTERCEPT_EXIT_PG_NONCPU_ENABLE: 0: State machine does not halt at the start of the non-CPU power ungating sequence 1: State machine halts at the start of the non-CPU power ungating sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
17	0x0	INTERCEPT_ENTRY_RG_CPU_ENABLE: 0: State machine does not halt at the start of the CPU rail gating sequence 1: State machine halts at the start of the CPU rail gating sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
16	0x0	INTERCEPT_EXIT_RG_CPU_ENABLE: 0: State machine does not halt at the start of the CPU rail ungating sequence 1: State machine halts at the start of the CPU rail ungating sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
15	0x0	INTERCEPT_ENTRY_PG_CORE0_ENABLE: 0: State machine does not halt at the start of the core0 power state entry sequence 1: State machine halts at the start of the core0 power state entry sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
14	0x0	INTERCEPT_EXIT_PG_CORE0_ENABLE: 0: State machine does not halt at the start of the core0 power state exit sequence 1: State machine halts at the start of the core0 power state exit sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
13	0x0	INTERCEPT_ENTRY_PG_CORE1_ENABLE: 0: State machine does not halt at the start of the core1 power state entry sequence 1: State machine halts at the start of the core1 power state entry sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
12	0x0	INTERCEPT_EXIT_PG_CORE1_ENABLE: 0: State machine does not halt at the start of the core1 power state exit sequence 1: State machine halts at the start of the core1 power state exit sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
11	0x0	INTERCEPT_ENTRY_PG_CORE2_ENABLE: 0: State machine does not halt at the start of the core2 power state entry sequence 1: State machine halts at the start of the core2 power state entry sequence. Issues interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
10	0x0	INTERCEPT_EXIT_PG_CORE2_ENABLE: 0: State machine does not halt at the start of the core2 power state exit sequence 1: State machine halts at the start of the core2 power state exit sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
9	0x0	INTERCEPT_ENTRY_PG_CORE3_ENABLE: 0: State machine does not halt at the start of the core3 power state entry sequence 1: State machine halts at the start of the core3 power state entry sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE
8	0x0	INTERCEPT_EXIT_PG_CORE3_ENABLE: 0: State machine does not halt at the start of the core3 power state exit sequence 1: State machine halts at the start of the core3 power state exit sequence. Issues an interrupt to ARM7/BPMP-Lite 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	INTERRUPT_PENDING_NONCPU: When set, it is an indication of an FC interrupt pending before starting non CPU Power Gating/Power Ungating. Sequence is halted until the bit is cleared. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING
6	0x0	INTERRUPT_PENDING_CRAIL: When set, it is an indication of an FC interrupt pending before starting CPU rail gating/ ungating. Sequence is halted until the bit is cleared. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING
5	0x0	INTERRUPT_PENDING_CORE0: When set, it is an indication of an FC interrupt pending from core0. Sequence is halted until the bit is cleared. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING
4	0x0	INTERRUPT_PENDING_CORE1: When set, it is an indication of FC interrupt pending from core1. Sequence is halted until the bit is cleared. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING
3	0x0	INTERRUPT_PENDING_CORE2: When set, it is an indication of FC interrupt pending from core2. Sequence is halted until the bit is cleared. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING
2	0x0	INTERRUPT_PENDING_CORE3: When set, it is an indication of FC interrupt pending from core3. Sequence is halted until the bit is cleared. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING
1	0x0	CC4_INTERRUPT_PENDING: Indicates that an interrupt was issued by FC to LIC after HVC state has entered. Interrupt stays asserted until ARM7/BPMP-Lite has cleared this bit. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING
0	0x0	HVC_INTERRUPT_PENDING: Indicates that an interrupt was issued by FC to LIC before starting HVC sequence or after exiting HVC sequence. Interrupt stays asserted until ARM7/BPMP-Lite has cleared this bit. This is set by hardware and cleared by software. Software should write "1" to this bit to clear it. 0 = NOT_PENDING 1 = PENDING

17.2.16 FLOW_CTLR_CC4_HVC_CONTROL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x0000000X (0b00000000000000000000000000000xx0)

Bit	R/W	Reset	Description
31:3	RW	0x0	HVC_RES_TIME_THRESHOLD: This represents the programmable time threshold required to enter HVC
2:1	RO	X	CC4_HVC_STS: This is a status register that indicates whether HVC entry/exit sequence has started/completed. 0 = CC4_HVC_NOT_ENTERED 1 = CC4_HVC_ENTRY_IN_PROGRESS 2 = CC4_HVC_EXIT_IN_PROGRESS 3 = CC4_HVC_ENTERED
0	RW	0x0	CC4_HVC_ENABLE: 0 = DISABLE 1 = ENABLE

17.2.17 FLOW_CTLR_CC4_RETENTION_CONTROL_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xx0)

Bit	Reset	Description
31:3	0x0	RET_RES_TIME_THRESHOLD: This represents the programmable threshold required to enter Retention in units of microseconds.

Bit	Reset	Description
0	0x0	CC4_RET_ENTERED: Indicates that CC4 Retention has been entered (VDD_CPU -> Vret). Set by BPMP_Lite at the end of the sequence 0 = CLEAR 1 = SET

17.2.18 FLOW_CTLR_CC4_FC_STATUS_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00x)

Bit	R/W	Reset	Description
2	RW	0x0	CC4_LIC_INTR_EN: This allows LIC interrupts to act as retention wake events. For HVC, this should be "0". On retention entry, BPMP can enable this after blocking interrupts to GIC. On retention exit, BPMP should clear this. 0 = DISABLE 1 = ENABLE
1	RW	0x0	CC4_WAKE_MASK: When set (by BPMP-Lite), prevents the FC from processing wake up events. It is used to allow the retention entry sequence to complete once it has reached certain point.
0	RO	X	INT_STATUS: Indicates that an interrupt to FC is pending in HVC. This register is updated by hardware and is readable by software. Software should read this only in HVC to decide whether to enter retention or not. 0 = PENDING 1 = NOT_PENDING

17.2.19 FLOW_CTLR_CC4_CORE0/1/2/3_CTRL_0

CC4_COREx_CTRL_0 offsets are as follows:

FLOW_CTLR_CC4_CORE0_CTRL_0: Offset: 0x6c

FLOW_CTLR_CC4_CORE1_CTRL_0: Offset: 0x70

FLOW_CTLR_CC4_CORE2_CTRL_0: Offset: 0x74

FLOW_CTLR_CC4_CORE3_CTRL_0: Offset: 0x78

Each of the four blocks has the same registers and bits.

Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:3	0x0	CORE_IDLE_TIMER: This register initializes a countdown timer that is used to determine whether to enter HVC or Retention. There is one for each core. The smallest of them is compared to a programmable residency threshold time (RET_RES_TIME_THRESHOLD for retention or HVC_RES_TIME_THRESHOLD for HVC). Its value represents units of microseconds. The countdown timer initialized by this register value counts down at every microsecond pulse. The hardware countdown timer is initialized to this value and enabled as soon as the core enters WFI (= asserts STANDBYWFI).
2	0x0	TIMER_COUNTDOWN_VALID: If this bit is set, the respective core is allowed to participate in the decision to enter CC3 HVC4 or CC4 Retention. If this bit is not set, the CORE_IDLE_TIMER does not countdown.
1	0x0	CORE_RET_ENABLE: If the respective TIMER_COUNTDOWN_VALID bit is set, the core participates in the decision to enter Retention and interrupt the BPMP-Lite. If this bit is not set, then the core cluster does NOT enter Retention and the BPMP-Lite is not interrupted. Software should initialize this bit every time before the core executes WFI (to enter C1 or C6) and software running on this specific core desires that the core cluster enters retention once the core has entered the requested power state (C1 or C6). It is reset when the core in C1 de-asserts STANDBYWFI (i.e., exiting C1) or when a wake-up event is received for the PG (in C6) core. It is also reset during an FC reset.
0	0x0	CORE_HVC_ENABLE: If the respective TIMER_COUNTDOWN_VALID bit is set, the core participates in the decision to enter HVC. If this bit is not set, the core cluster does NOT enter HVC. Software should initialize this bit every time before the core executes WFI (to enter C1 or C6) and software running on this specific core desires that the core cluster is allowed to enter HVC once the core has entered the requested power state (C1 or C6). It is reset when the core in C1 de-asserts STANDBYWFI (i.e., exiting C1) or when a wake-up event is received for the power-gated (in C6) core. It is also reset during an FC reset.

17.2.20 FLOW_CTLR_CORE0/1/2/3_IDLE_COUNTER_0

COREx_IDLE_COUNTER_0 offsets are as follows:

FLOW_CTLR_CORE0_IDLE_COUNTER_0: Offset: 0x7c

FLOW_CTLR_CORE1_IDLE_COUNTER_0: Offset: 0x80

FLOW_CTLR_CORE2_IDLE_COUNTER_0: Offset: 0x84

FLOW_CTLR_CORE3_IDLE_COUNTER_0: Offset: 0x88

Each of the four blocks has the same registers and bits.

Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:3	X	CORE_IDLE_COUNTER: This register captures the current value of the core's idle countdown counter which is initialized to CC4_CORE<x>_CTRL.CORE_IDLE_TIMER at STANDBYWFI asserted by the specific core.

17.2.21 FLOW_CTLR_CC4_HVC_RETRY_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	THRESHOLD: This represents the threshold to retry HVC entry after QDENY signals are asserted.

17.2.22 FLOW_CTLR_L2FLUSH_TIMEOUT_CNTR_0

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	STATUS: This bit is set by hardware in case of l2flushrequest timeout. This bit can be cleared by software. Software should write "1" to this bit to clear it. Hardware clears this before starting next Rail Gate request or non CPU PG request. 0 = CLEAR 1 = SET
30:1	0x0	THRESHOLD: l2flush timeout counter starts incrementing after l2flushrequest is asserted. It asserts timeout once the counter reaches the CNTR_THRESHOLD value. If l2flushdone signal is not asserted for CNTR_THRESHOLD # of sclks after l2flushrequest is asserted, the flow controller de-asserts the l2flushrequest signal, asserts the STATUS field, and will not proceed with Rail Gate/Non-CPU PG request
0	0x0	ENABLE: Enables L2FLUSH TIMEOUT counter 0 = DISABLE 1 = ENABLE

17.2.23 FLOW_CTLR_L2FLUSH_CONTROL_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx01)

Bit	Reset	Description
1	0x0	REQ: Issues software L2 FLUSH request. Software sets this bit to '1' to initiate L2 FLUSH request. The flow controller sends a software L2FLUSH request only if all unpower-gated cores are in WFX and the REQ bit is set. Hardware clears this bit when the software triggered L2 FLUSH request is done. In case of software timeout, software should explicitly clear this bit. L2FLUSH_TIMEOUT_CNTR is only used for hardware flush. 0 = DISABLE 1 = ENABLE
0	0x1	ENABLE: Enables hardware L2FLUSH handshake 0 = DISABLE 1 = ENABLE

17.2.24 FLOW_CTLR_BPMP_CLUSTER_CONTROL_0

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	ACTIVE_CLUSTER_LOCK: This field can only be written by a secure write. The ACTIVE_CLUSTER field can only be written if the ACTIVE_CLUSTER_LOCK bit is "0". 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	CLUSTER_SWITCH_ENABLE: This field should only be written by the BPMP prior to allowing master core initiate CC6. 0 = DISABLE 1 = ENABLE
0	0x0	ACTIVE_CLUSTER: This field should only be written by the BPMP during cluster switch sequence after CPU cluster enters CC6. ACTIVE_CLUSTER field can only be written if ACTIVE_CLUSTER_LOCK bit is "0". 0 = FAST 1 = SLOW

CHAPTER 18: MEMORY CONTROLLER

18.1 Introduction

Tegra® X1 devices feature two Memory Controllers and two External Memory Controllers, one set per 32-bit channel.

The Tegra X1 memory controller (MC) handles memory requests from internal clients and arbitrates among them to allocate memory bandwidth for DRAM. The external memory controller (EMC) communicates with external LPDDR3 or LPDDR4 devices.

Key features in the Tegra X1 memory controller include:

- Enhanced arbiter design for higher memory efficiency
- Dual channels
- System Memory Management Unit (SMMU)/Translation Unit (TU) for virtual to physical address mapping for any device
- Support for LPDDR3 and LPDDR4 SDRAMs
- 64-bit memory interface implemented as 2 32-bit channels (with x16 subpartitions for LPDDR4)
- 34-bit virtual addressing
- Up to 8 GB memory capacity
- Swiss-Cheese mode beyond 4GB
- Support for two DRAM ranks of unequal size and unequal device density
- Generalized security apertures
- Variable transaction sizes based on the requests from the clients (one 64-byte transaction with variable dimensions, two 32-byte transactions with variable dimensions, etc.)

The memory interface speed varies with memory type and the specific Tegra X1 SKU, so is not stated in this document.

Terminology Used in this Section

Term	Definition
Channel	Independent memory controller (including row-sorter and EMC) with dual sub-partitions * 32b Channel for LPDDR4
DRAM Channel	Minimum width independent port supported by DRAM * 16b data width on LPDDR4 Matches up to one of our subpartitions
Subpartition	½ the data width of a memory controller channel. The memory controller provides a single read or write command and row address to both subpartitions in the channel to transfer 64B, but provides 3 independent column address bits to each subpartition, allowing it access different 32B sectors of a GOB between the subpartitions. * 16b Subpartition for LPDDR4

18.2 Memory Controller Architecture

The memory controller architecturally consists of the following parts:

- Arbitration Domains (ADs), which can handle a single request or response per clock from a group of clients. Typically, a system has a single Arbitration Domain, but an implementation may divide the client space into multiple Arbitration Domains to increase the effective system bandwidth. Multiple Traffic Classes within a single Arbitration Domain and Protocol Arbiter are allowed.

- Protocol Arbiters (PAs), which manage a related pool of memory devices. A system may have a single Protocol Arbiter or multiple Protocol Arbiters.
- Memory Crossbar, which routes request and responses between Arbitration Domains and Protocol Arbiters. In the simplest version of the system, the Memory Crossbar is a pass through between a single Arbitration Domain and a single Protocol Arbiter.
- Global Resources, which include entities such as configuration registers which are shared across the Memory Subsystem.
- Write CAMs (WCAMs), which improves performance and throughput for PCIe® ordered clients (PCIe, SATA, HDA, and USB3), CPU writes, and CPU copies.
- Memory Controller Client Interface (MCCIF), which provides a standardized interface for access to the Memory Controller.
- Translation Unit, which handles virtual-to-physical address translation, aperture decode, physical address security checks, and protocol arbiter-specific decodes (such as external DRAM address decodes).

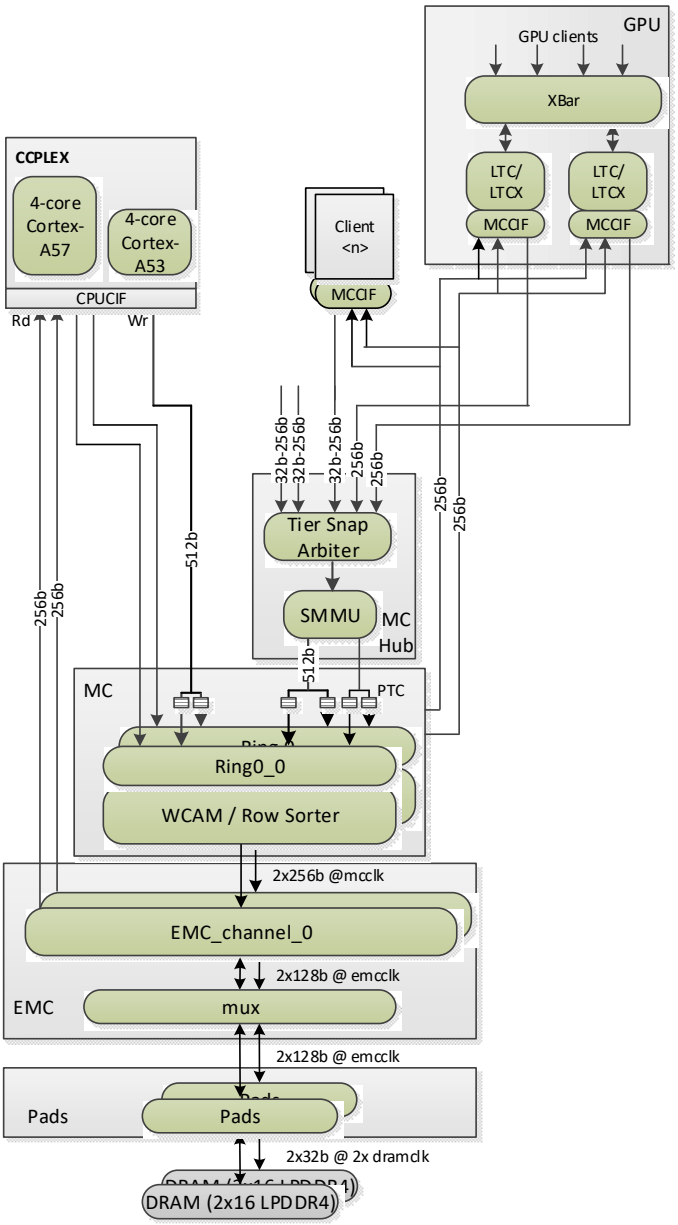
The Tegra X1 device supports either a single or a dual channel memory interface depending on the memory type and width.

There is an alternate path through the memory controller to access IRAM via AHB re-direction.

The figure below is a view of how memory requests are arbitrated in the Tegra X1 MSS. Memory client requests are first arbitrated through a sequence of ring arbiters which perform a type of round-robin arbitration. There are three ring arbiters referred to as ring0, ring1, and ring2. Ring1 arbiter clients are the ISO clients (display and camera) and the winner of the ring2 arbiter. Each ring has a rate control mechanism referred to as Priority Tier Snap Arbiter (PTSA). The client’s bandwidth guarantee is specified by the PTSA rate (also referred to as “DDA”).

The block labelled “Row Sorter” is a pending request buffer (it sorts requests by the DRAM row that it refers to). This row sorter is made up of many “bank queues” which hold the requests made to the same DRAM bank/row. The number of requests pending in the row sorter can affect whether ring1 or ring2 arbiters are throttled (slowed down) based on thresholds.

Figure 34: Memory Request Arbitration



18.3 Hardware Features

18.3.1 LPDDR4 Support

LPDDR4 DRAMs differ from LPDDR3 in the following significant ways:

- Up to 2x the bandwidth per pin
- 2x the DRAM clock frequency
- Burst length of 16 (compared to burst length of 8 on LPDDR3)
- 32-bit device composed of two independent 16-bit channels (64-bit LPDDR4 has four independent channels)
- Initialization time training is required for frequencies above 400 MHz. Periodic/DVFS write training for temperature compensation is needed above 1000 MHz.
- Has optional internal ECC, which extends time for masked writes

- Has SDR address and control

A 16-bit LPDDR4 channel with burst length 16 has the same 32B DRAM atom as a 32-bit LPDDR3 channel with burst length 8. The 64-bit LPDDR interface is treated as two 32-bit channels, each with two 16-bit sub-channels (or sub-partitions). Each 32-bit LPDDR4 channel is logically the equivalent of a 64-bit Tegra K1 LPDDR3 channel.

18.3.2 LPDDR4 Masked Writes

On LPDDR4, with internal ECC, masked writes are internally converted to read-modify-writes and are 4x slower than normal writes (tCCDMW = 4 * tCCD). This timing is used from any Write (masked/unmasked) to a Masked Write on the same bank. There is no penalty across banks or for any other commands.

The arbitration is updated to correctly track the timing for masked writes, and will not allow masked writes to be part of the cycle's arbitration if they do not meet tCCDMW. This timing will be tracked per-bank and per-rank, so it will only mask the bank that does not meet the timing. Other banks will still be allowed to issue their requests.

64B write requests are masked writes if any byte is not enabled. 16B write requests are always masked.

32B requests are more complicated. They are masked if any byte for the 32B is not enabled. They are also masked if the AP (auto-precharge) bit is set to allow the page to be closed in both subpartitions. Otherwise 32B requests are not masked writes. When AP is set, the masked write request is sent to both subpartitions. Otherwise, it is only sent to the target subpartition for the 32B request.

18.3.3 Rank 1 to Rank 0 Support

The industry standard is to map the larger DRAM rank to CS1. For addressing simplicity, the larger rank starts at DRAM address 0; the hole from the smaller rank is at the top of address space. A programmable swizzle of per-rank command bits is implemented between the EMC and the control pads. The swizzle registers are documented in the "Memory Controller Registers" section below.

Note: *In Tegra X1 devices, the DRAM configuration must be identical across channels.*

18.3.4 Channel Interleave and Swizzle

The channel interleave is determined by the MC_EMEM_ADR_CFG_CHANNEL_MASK_0 register. The following equation determines the channel:

```
channel = ^(EMEM_ADR_CFG_CHANNEL_MASK_0_EMEM_CHANNEL_MASK[31:9] & addr[31:9])
```

Mask bits [10:9] select single channel vs. dual-channel modes and 1KB interleave, each with its own physical address to <channel, device, row, bank, column> mapping.

Interleave	MASK[10:9]
Single channel	2'b00
1KB	2'b10

The basic interleave (1KB) is determined by mask bits [10:9]. To avoid channel camping, address MSBs need to be hashed into the channel field (controlled by the MSBs of the mask register).

Recommended settings of the register are the following:

Interleave	MASK[31:0]
Single channel	32'b 0000 0000 0000 0000 0000 000x xxxx xxxx
1KB	32'b 1111 1111 1111 1111 0010 010x xxxx xxxx

In single channel mode, the channel bit must always be zero. For the dual-channel cases, the address space is divided into a locally "manicured region" of 128KB (256 pixels wide by 128 pixels tall, assuming the recommended block height of 16 gobs). In

this manicured region, the channels are interleaved in a regular checkerboard. In 512B mode, the tile size is 64Bx8. In 1KB mode, the tile size is 64Bx16. With the recommended block height of 16 gobs (128 lines), having mask bit 13 set mirrors the channels in 8 successive columns. At the 64KB boundary, the channel mapping should be inverted to dovetail with the 128KB bank manicure described in the next section. Beyond 128KB randomization of the channel is desired, so the system will not have performance divots at particular strides, which is done by XORing in the address MSBs. Note that the manicured region will result in the same channel being hit 4 times in a row if the stride is exactly 16KB.

18.3.5 Bank Interleave and Swizzle

A similar approach is taken to load balance across banks: the banks are interleaved in a manicured region of 64KB (single-channel case) or 128KB (dual-channel case). Additionally XORed are a hash of address MSBs to avoid systematic abutment issues between manicure tiles or bank camping when accessing with a constant stride.

The bank interleave formula, assuming 3 bank bits, is as follows:

```
bank[2:0] = msb_swizzle[2:0] ^ manicure_swizzle[2:0] ^ bank_base[2:0];
```

The terms in the equation are:

- bank_base. This is the raw bank number from the physical address (typically a one-hot-encoding of the bank bits)
- manicure_swizzle. This swizzles the bank bits within the manicure region to give an ideal interleave of banks within the region.
- msb_swizzle. This is a hash of address MSBs that is constant for each manicure region but random from one manicure region to the next, with special care taken to prevent horizontal abutment conflicts.

Single-Channel Settings

For the single-channel case, the recommended bank mask settings are the following:

```
bank_base[2:0]        = addr[12:10]
manicure_swizzle[2:0] = {addr[15]^addr[13],addr[14],addr[13]}
Galois[16:31][2:0]    = {5,7,3,6,7,5,1,6,5,1,6,5,4,3,5,1}
EMEM_ADR_CFG_BANK_MASK_0_EMEM_BANK_MASK = 0b 1110 1011 0111 0111 0010 01xx xxxx xxxx
EMEM_ADR_CFG_BANK_MASK_1_EMEM_BANK_MASK = 0b 0010 0100 1001 1110 0100 10xx xxxx xxxx
EMEM_ADR_CFG_BANK_MASK_2_EMEM_BANK_MASK = 0b 0101 1101 1011 1011 1011 00xx xxxx xxxx
```

Dual-channel settings

For the dual-channel case, the recommended bank mask settings are the following:

```
bank_base[2:0]        = addr[13:11] = xaddr[12:10]
manicure_swizzle[2:0] = {0,addr[14],addr[15]} = (0,xaddr[13],xaddr[14])
Galois[17:31][2:0]    = {1,3,5,4,7,2,3,4,6,5,1,7,2,3,5,0}

EMEM_ADR_CFG_BANK_MASK_0_EMEM_BANK_MASK = 0b 0110 1110 0101 0111 0100 01xx xxxx xxxx
EMEM_ADR_CFG_BANK_MASK_1_EMEM_BANK_MASK = 0b 0011 1001 0111 0010 0010 10xx xxxx xxxx
EMEM_ADR_CFG_BANK_MASK_2_EMEM_BANK_MASK = 0b 0100 1011 1001 1100 0001 00xx xxxx xxxx
```

18.3.6 Hub + Dual-channel Architecture

To support LPDDR4, which has 2x the bandwidth per pin of LPDDR3, Tegra X1 devices provide two MC channels. The MC (shown in Figure 34) is configured as follows:

- A central hub (MCHUB), containing the snap arbiter tree and SMMU
- Two MC channels, each with an instance of ring 0, the WCAM, and row sorter
- Two EMC channels and EMC

Each channel arbitrates and schedules requests independently. When driving LPDDR4, each of the two channels drives 32 DRAM I/Os with independent arbitration and scheduling. The DRAM configuration must be identical across the two channels.

18.3.6.1 Cross Channel Ordering

Ordering is more complex on Tegra X1 devices because of its two channels. There is one WCAM instance per channel, so there is no single point of global visibility.

WAW Ordering Via Write-Ack

Use cases that use write-ack ensure visibility of writes before issuing dependent work. The WCAM for each channel returns a write_ack only when the write data will be visible to subsequent reads to that channel. For clients with a single write request interface, the MC reorders write_acks between the channels, so a write_ack is only returned to the client if write_acks for all previous requests (regardless of the channel). So when a write_ack for a request has been received, it is guaranteed that all prior writes, regardless of the channel, are globally visible.

The CPU has separate write interfaces for each channel and has a separate write_ack interface for each channel. The CPU is responsible for proper ordering of its requests.

The GPU has separate write clients for each L2 slice, which have no ordering requirements between them, so the lack of ordering for writes between the two GPU clients does not introduce new ordering requirements to the GPU. The GPU uses explicit membars and sysmembars to ensure writes have been appropriately flushed.

WAW Ordering Using the Producer-Consumer Model

In the producer/consumer model, a client issues data writes followed by a flag write. A consumer client reads the flag and when it changes, assumes the data is valid. A data write from the producer could pass the channel fork and be stalled above WCAM for that channel while the flag write goes down the other channel and could be viewed by the consumer, who would believe the data is valid. Although this is pathological, certain clients, such as PCIe-ordered clients and CPU strongly-ordered and dev accesses require strict ordering enforcement.

The CPU enforces its own ordering for strongly-ordered and dev accesses. A strongly ordered or dev request to a particular channel will not be issued (from CPUCIF to MC) until all prior writes to other channel have been acknowledged.

The MC enforces ordering for ordered clients such as PCIe and other clients that require PCIe ordering. It does this with an interlock at each PCFIFO. The enable for the interlock is based on client-ID, and will be programmable by config registers. The config register should follow the same formatting as the SMMU Ordered config bits.

The interlock contains a register that stores the client address bits (down to the channel bit based on the current channel interleave) for unacknowledged writes from the client and a counter that tracks how many unacknowledged writes are pending. This circuit will exist on the output of all PCFIFOs. Reads are always allowed (there are no RAR ordering requirements from the MC), but writes will be blocked if the counter is non-zero and the address of the new request does not match the stored address. Note that this circuit is conservative: it interlocks writes that cross a potential channel interleave boundary (1KB), but the channel may not toggle on every boundary.

WAW Ordering for AHB Clients

The AHB client is a special case, since it conveys transactions on behalf of multiple internal clients that can be active on the AHB bus simultaneously. The aggregate AHB bandwidth requirement is 520 MB/s at 204 MHz emcclk. Assume that bandwidth is evenly divided between two clients, with each client issuing a 64B request (which is really a 2x-inflated 32B request) every 64B/(260 MB/s) = 0.25 µs. Assume 100 emcclks of round-trip write ack latency (this is the conservative estimate for non-DRAM-limited cases). Because requests from different clients will almost certainly be to different DRAM pages, each request will incur an interlock stall of 100 emcclks / 204 MHz = 0.5 µs. So indiscriminately stalling could throttle bandwidth by a factor-of-2 from the bandwidth requirement. In practice, it will be unlikely for two clients to be equally matched in bandwidth. Also, the write ack latency will typically be lower, assuming a good ratio of TLB hits to misses.

Clients with Internal Write-after-Write Ordering Barriers

Several clients that require PCIe ordering, such as PCI and SATA, have a shim that can enable a write-after-write ordering barrier for a programmable block size. For clients other than CPU and AHB, these are recommended to be disabled and the PC-FIFO barrier described above be used instead, since write-ack latency is shorter from the PCFIFO and performance will therefore be better.

18.3.6.2 PTSA Register Programming for Dual Channel

The ring1 and ring2 PTSA DDAs see full memory bandwidth, but each ring0 instance sees only half of memory bandwidth. PTSA registers need to be programmed to take this into account:

- **GRANT_DECREMENT.** GRANT_DECREMENT is set to allow the maximum bandwidth required for any client at any arbiter. Each ring1 feeder carries only half the ring1 bandwidth—at most 0.50% of peak DRAM bandwidth.
- **DDA rate settings at ring 0.** These should be scaled by 0.5 in dual-channel mode, since there are two parallel instances of ring 0.

18.3.6.3 Dual-Channel Registers

Some parts of the design are replicated per-channel, while others are not. Configuration registers are sometimes duplicated for the two channels, while others are not, and have a single instance in the design.

Configuration registers live in 6 different places. MCHUB, MC Channel (0+1), EMC Channel (0+1), and the EMC pad macros. There can be duplicated registers across channels.

Accesses to MC and EMC should normally be done through the broadcast range. The only exception is the per-channel statistics registers. When using unicast accesses, it is possible to configure MC+EMC incorrectly, since the common areas (MCHUB and EMC Pad Macros) are available through all apertures.

There are 6 configuration apertures in MC+EMC:

Aperture	Write Destination	Write Action	Read Source
MC = MCB (broadcast)	MCHUB + MC Channel 0 + MC Channel 1	Write a Reg. in MCHUB if offset matches and discard if does not. Write a Reg. in MC0 if offset matches and discard if does not. Write a Reg. in MC1 if offset matches and discard if does not.	MCHUB +MC Channel 0
MC0 (unicast)	MCHUB + MC Channel 0	Write a Reg. in HUB if offset matches and discard if does not. Write a Reg. in MC0 if offset matches and discard if does not.	MCHUB + MC Channel 0
MC1 (unicast)	MCHUB +MC Channel 1	Write a Reg. in HUB if offset matches and discard if does not. Write a Reg. in MC1 if offset matches and discard if does not.	MCHUB + MC Channel 1
EMC = EMCB (broadcast)	Pad Macros + EMC Channel 0 + EMC Channel 1	Write a Reg. in Pad Macros if offset matches and discard if does not. Write a Reg. in EMC0 if offset matches and discard if does not. Write a Reg. in EMC1 if offset matches and discard if does not.	Pad Macros + EMC Channel 0
EMC0 (unicast)	Pad Macros + EMC Channel 0	Write a Pad Macros if offset matches and discard if does not. Write a Reg. in EMC0 if offset matches and discard if does not.	Pad Macros + EMC Channel 0
EMC1 (unicast)	Pad Macros + EMC Channel 1	Write a Reg. in Pad Macros if offset matches and discard if does not. Write a Reg. in EMC1 if offset matches and discard if does not.	Pad Macros + EMC Channel 1

18.3.6.4 ARC Requests

All requests are checked against the IRAM aperture at the output of ring1. Any requests inside IRAM are then routed to ARC. Responses for ARC requests are always merged in the MC channel 0 response stream.

ARC responses are always sent back with rot==1 in order to properly merge ARC requests in the MCCIF during dual channel operation.

18.3.6.5 Interrupt Handling

Interrupts need to be handled in a backwards compatible way. Interrupts and decode error status signals are routed across channels to allow all interrupt information to be accessible from channel 0.

There are 2 modes of operation:

- Combined interrupt mode, where the 2 channels are combined into a single set of interrupt registers (in channel 0).
- Independent interrupt mode combines the actual interrupt signals, but none of the status registers are merged.

The local interrupt status signals need to be updated to trigger when the other channel's interrupt is set, not off the other channel's interrupt status signals, since the interrupt clear might not happen in the two channels at the same exact time.

Combined Interrupt Mode

Some characteristics of combined interrupt mode include:

- MC Channel 0's interrupt registers report interrupts for both channel 0 and channel 1
- MC Channel 0's interrupt registers include SMMU interrupts
- There is only a single set of registers to read during an MC interrupt:
  - Read INTSTATUS using broadcast read
  - Clear INTSTATUS using broadcast write
  - Read interrupt error information registers using broadcast read

Independent Interrupt Mode

In independent interrupt mode, each MC channel independently tracks its own interrupt information. The interrupt sent to the core is an OR of the two channels. **This is only intended as a debug mode for MC.**

- Software needs to read INTSTATUS from both channels using unicast reads
- Software can clear INTSTATUS using unicast or broadcast writes
- Requires software to know if there are
- Allows tracking interrupts independently in each channel
- Both channels will present SMMU interrupts

18.3.6.6 MC Cross Channel Interrupt Signals

These should always be merged into channel 0 when operating in the combined interrupt mode, except channel2mc_intr, which should only be merged in when not operating in combined interrupt mode.

These signals should be tied to 1'b0 on channel 1.

Port Signal	Source signal	Config field
channel2mc_intr	hrd_mc2host1x_intr	N/A

Port Signal	Source signal	Config field
channel2mc_decerr_emem_trigger	flopped tu2reg_decerr_emem * But don't trigger on SMMU page fault	MC_INTSTATUS_0_DECERR_EMEM_INT_FIELD
channel2mc_decerr_mts_trigger	flopped tu2reg_decerr_mts	MC_INTSTATUS_0_DECERR_MTS_INT_FIELD
channel2mc_decerr_vpr_trigger	flopped tu2reg_decerr_vpr	MC_INTSTATUS_0_DECERR_VPR_INT_FIELD
channel2mc_decerr_sec_trigger	flopped tu2reg_secerr_sec	MC_INTSTATUS_0_SECERR_SEC_INT_FIELD
channel2mc_security_violation_trigger	flopped tu2reg_security_violation	MC_INTSTATUS_0_SECURITY_VIOLATION_INT
channel2mc_arbitration_emem_trigger	flopped earb2reg_arbitration_emem	MC_INTSTATUS_0_ARBITRATION_EMEM_INT
channel2mc_decerr_generalized_trigger	flopped tu2reg_decerr_generalized	MC_INTSTATUS_0_DECERR_GENERALIZED_CARVEOUT_INT_FIELD
channel2mc_err_adr	tu2reg_err_adr	MC_ERR_ADR_0_ERR_ADR
channel2mc_err_adr1	tu2reg_err_adr1	MC_ERR_STATUS_0_ERR_ADR1
channel2mc_err_adr_hi	tu2reg_err_adr_hi	MC_ERR_STATUS_0_ERR_ADR_HI
channel2mc_err_id	tu2reg_err_id	MC_ERR_STATUS_0_ERR_ID
channel2mc_err_rw	tu2reg_err_rw	MC_ERR_STATUS_0_ERR_RW
channel2mc_err_swap	tu2reg_err_swap	MC_ERR_STATUS_0_ERR_SWAP
channel2mc_err_type	tu2reg_err_type	MC_ERR_STATUS_0_ERR_TYPE
channel2mc_err_security	tu2reg_err_security	MC_ERR_STATUS_0_ERR_SECURITY
channel2mc_err_(mts|sec|vpr|generalized)_adr	tu2reg_err_(mts|sec|vpr|generalized)_adr	MC_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_ADR_0_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_ADR
channel2mc_err_(mts|sec|vpr|generalized)_adr1	tu2reg_err_(mts|sec|vpr|generalized)_adr1	MC_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_STATUS_0_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_ADR1
channel2mc_err_(mts|sec|vpr|generalized)_adr_hi	tu2reg_err_(mts|sec|vpr|generalized)_adr_hi	MC_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_STATUS_0_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_ADR_HI
	tu2reg_err_(mts|sec|vpr|generalized)_id	MC_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_STATUS_0_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_ID
	tu2reg_err_(mts|sec|vpr|generalized)_rw	MC_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_STATUS_0_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_RW
	tu2reg_err_(mts|sec|vpr|generalized)_swap	MC_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_STATUS_0_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_SWAP
	tu2reg_err_(mts|sec|vpr|generalized)_security	MC_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_STATUS_0_ERR_(MTS|SEC|VPR|GENERALIZED_CARVEOUT)_SECURITY
channel2mc_err_generalized_aperture_id	tu2reg_err_generalized_aperture_id	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_APERTURE_ID

Port Signal	Source signal	Config field
channel2mc_err_generalized_security	tu2reg_err_generalized_security	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_SECURITY
channel2mc_err_generalized_carveout_aperture_id	tu2reg_err_generalized_carveout_aperture_id	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_CARVEOUT_APERTURE_ID
channel2mc_err_generalized_chk_aperture_id	tu2reg_err_generalized_chk_aperture_id	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_CHK_APERTURE_ID
channel2mc_err_generalized_chk_aperture	tu2reg_err_generalized_chk_aperture	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_CHK_APERTURE
channel2mc_err_generalized_chk_adr_type	tu2reg_err_generalized_chk_adr_type	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_CHK_ADR_TYPE
channel2mc_err_generalized_chk_aperture_id_external	tu2reg_err_generalized_chk_aperture_id_external	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_CHK_APERTURE_ID_EXTERNAL
channel2mc_err_generalized_chk_gpu_access	tu2reg_err_generalized_chk_gpu_access	MC_ERR_GENERALIZED_CARVEOUT_STATUS_0_ERR_GENERALIZED_CARVEOUT_CHK_GPU_ACCESS

MC Hub to Channel Interrupt Signals

These should always be merged into channel regardless of the interrupt mode. Route to both MC channels.

Port Signal	Source signal	Config field
mchub2mc_invalid_apb_asid_update	flopped tu2reg_invalid_apb_asid_update	MC_INSTATUS_0_INVALID_APB_ASID_UPDATE_INT
mchub2mc_invalid_smmu_page	flopped tu2reg_invalid_smmu_page	MC_INTSTATUS_0_INVALID_SMMU_PAGE_INT
mchub2mc _err_adr	invalid_smmu_page_adr	MC_ERR_ADR_0_ERR_ADR
mchub2mc_err_adr1	invalid_smmu_page_adr1	MC_ERR_STATUS_0_ERR_ADR1
mchub2mc_err_adr_hi	invalid_smmu_page_adr	MC_ERR_STATUS_0_ERR_ADR_HI
mchub2mc_err_id	invalid_smmu_page_id	MC_ERR_STATUS_0_ERR_ID
mchub2mc_err_rw	invalid_smmu_page_rw	MC_ERR_STATUS_0_ERR_RW
mchub2mc_err_swap	invalid_smmu_page_swap	MC_ERR_STATUS_0_ERR_SWAP
mchub2mc_err_type	invalid_smmu_page_type	MC_ERR_STATUS_0_ERR_TYPE
mchub2mc_err_security	invalid_smmu_page_security	MC_ERR_STATUS_0_ERR_SECURITY
mchub2mc_err_page_nonsecure	invalid_smmu_page_nonsecure	MC_ERR_STATUS_0_ERR_INVALID_SMMU_PAGE_NONSECURE
mchub2mc_err_page_writable	invalid_smmu_page_writable	MC_ERR_STATUS_0_ERR_INVALID_SMMU_PAGE_WRITABLE
mchub2mc_err_page_readable	invalid_smmu_page_readable	MC_ERR_STATUS_0_ERR_INVALID_SMMU_PAGE_READABLE

18.3.7 DRAM Protocol Arbiter Features

- Hybrid 2x32-bit / 1x64-bit data bus
  - 4 chip selects
  - 4 individually controllable clock-enables
  - Operates in either single x32 or hybrid 2x32 / 1x64 configuration
  - Supports per-byte data masks
- Each chip select may support different sizes and geometries
  - Size of Rank 0 must be larger than or equal to Rank 1
  - 2 or 3 bank bits
  - Column width: 9 to 12 bits
  - Row width: 12 to 16 bits
- Per-byte data masks
- Support for BL8
- Data transfer minimization:
- Support for 1T and 2T
- Support for low-power modes:
  - Software controllable entry/exit from: self-refresh, power down, deep power down
  - Hardware dynamic entry/exit from: power-down, self-refresh
  - Support for intermittent or disabled DLL
  - Disable unused address/command taps based on whether in POP or Discrete mode.
  - Pads use DPD mode during idle periods
- For Tegra X1 devices, trims can be set/adjusted per-byte AND per-chip-select. The PVT-compensated value can still be used and a PVT/frequency-compensated adjustment can be added to it (1/16, 1/8, 3/16 of a cycle) or a non-PVT-offset can be added.
- Support for x32, x16, or x8 chips attached to the channel
  - DQ/DQS swizzling: MRR_BYTESEL need to match byte swizzling.
- Deadline-based arbitration with a latency allowance that can be specified per-client, and under some circumstances, dynamically adjusted for a given client.

18.3.8 Memory Crossbar

The Memory Crossbar (MXB) is in charge of adapting Arbitration Domains (ADs) to Protocol Arbiters (PAs). In the simplest system with a single AD and PA running off of a single clock, the MXB is a simple pass-through or does not exist. The Tegra X1 MXB has one AD and two PAs, one for DRAM and the other for requests redirected to the AHB. The MXB handles conflict arbitration between responses from multiple PAs to the same AD.

18.3.9 Memory Sub-Partitions

The Tegra X1 Memory Controller implements sub-partitions. The 64 B memory atom is divided into two sectors: one sector containing bytes 0 to 31 and the other containing bytes 32 to 63. Each sector is stored in one of the two DRAM channels. With three additional column address bits for the second sector, clients can make 64 B requests of different shapes while retaining the bandwidth and power efficiency of a 64B memory atom. This feature also allows the CPU to request that one of the two 32 B sectors in a 64 B atom be returned first.

The sub-partition mechanism is used by the GPU, display clients (for rotation), VIC (for rotation), NVENC (in x,y flip mode), and the CPU (for critical-word-first). Clients that choose not to use the sub-partition feature set the extra column address bits to the same values as the other channel. They see the DRAM as a simple 64 B memory system. The sub-partition feature is only beneficial in systems with a 64 b memory. In systems with 32b memory, client addressing and request granularity are unchanged, but the MC treats each 32 B sector as an independent 32 B atom.

See the Section 18.7.2: Sub-Partitions for more information.

18.3.10 Global Resources

The Memory Subsystem also contains some global resources not owned by a particular AD or PA:

- A combined SMMU/TU
- Configuration registers and statistics counters

18.3.11 Supported Page Sizes

The MSS considers the following uses of pages and page size:

- DRAM page sizes from 1 KB to 4 KB.
- OS page: 4 KB for all operating systems
- Row Sorter page: 1 KB

18.4 Software Features

The majority of software interaction with the memory subsystem involves:

- Initial configuration
- Power management
- Device management
- Translation management
- ISO client bandwidth allocation
- Statistics and debugging

18.4.1 Initial Configuration

The details for configuring the Memory Subsystem (MSS) can be broken down into the following:

- Global Memory Subsystem configuration
- Arbitration Domain configuration
- Protocol Arbiter configuration
- Client configuration

Note: *This configuration must be performed as part of the chip-wide boot sequence*

18.4.1.1 Global Memory Subsystem Configuration

Some parts of the Memory Subsystem must be configured before any transfers are allowed into the system. The address map must be configured to set which portions of the physical address map are allocated to which Protocol Arbiters, and what portions of it are protected by the physical address protection mechanisms.

The Arbitration Domains operate off of a unified arbitration clock. This clock is further divided-down to produce a clock-rate independent clock for counting latency intervals across the Memory System (also known as “ticks”). This divide-down should be

programmed to a constant interval at initialization (and any other clock-rate change). A 30 ns interval is suggested since it is an even divide-down of many common DRAM clocks, but any convenient granularity may be chosen.

There are several global memory system tuning options that tend to shape the memory performance. Some of these can be set statically, at initialization time, others depend on clock frequencies. For example, with SMMU configuration, the SMMU requires software to set up and maintain page tables in memory, enable translation for clients, and assign clients to address space identifiers.

18.4.1.2 Arbitration Domain Configuration

Physical implementation decisions (such as the mapping of clients to partition clients) and default client bandwidth allocations require configuration options in the Arbitration Domain.

18.4.1.3 Protocol Arbiter Configuration

Each Protocol Arbiter has its own requirements for initialization. Timing parameters specific to the Arbiter, the DRAM and the current operating clock speed have to be written into Protocol Arbiter configuration registers. The type and geometry of the attached DRAM also have to be programmed. Afterwards, the DRAM will require a set of initialization cycles to be issued.

Device/Rank Geometry

DRAM devices come in many sizes, widths, etc. The arbiter must be programmed to drive the correct combination of address bits, data bits, and protocol.

Device Timing and Arbiter Timing

A DRAM arbiter has configuration related to timing parameters specific to the attached device. The JEDEC timing specifications for the device have to be converted from nanoseconds to cycle counts. The controller must also be programmed with the cycles per tick number for their particular clock domain (refer to Section 18.4.1.1: Global Memory Subsystem Configuration).

Other Arbiter Parameters

Other parameters related to arbitration that need to be configured include:

- overall number of requests outstanding
- which clients are considered isochronous
- which clients participate in power-saving hysteresis operations

Device/Rank Initialization and Calibration Cycles

In particular, PAs need to write registers in the DRAM devices via special cycles to initialize the DRAM. In general all special cycles to a device are done via a hardware/software handshake that looks like:

- Software writes information about the special cycle to an Arbiter-specific register
- The Controller does the special cycle out to the DRAM device(s) when timing requirements are met
- Software checks a status register to ensure the special cycle is complete; or for Mode Register Read operations, the Controller issues an interrupt to indicate that the special cycle has been consumed

Special cycles may be emitted during normal operation as well; the Arbiter will inject them into the data stream at the appropriate places (typically along with refresh cycles).

During device initialization, software must ensure that no client can access the memory. This can be done with per-module flush-enable bits.

18.4.1.4 Client Configuration

The Priority Tier Snap Arbiters normally are configured dynamically to provide bandwidth guarantees to ISO clients. However, for certain clients the PTSA settings are programmed statically (for example, when the PTSA is used as a timeout). For these clients, the PTSA settings are part of the initial configuration.

Latency allowance may also be configured dynamically for certain clients. Latency allowance should be programmed to default values during initial configuration.

18.4.2 Power Management

The primary forms of system power management involve transition to and from low-power DRAM states and management of clock frequency.

18.4.2.1 Low-Power States

The DRAM controller and DRAMs can operate in a number of low-power states. The controller can be programmed to automatically enter a precharge power-down or self-refresh state after some amount of time without a request. The power-down state can be independently entered for each individual DRAM device/rank – it is possible for one device/rank to be in power-down while another is active. If software knows that the system will not be making further DRAM requests, it can cause an immediate transfer to the self-refresh state by writing an override register. If multiple chip selects are available for the DRAM controller, both the hardware and software may choose to enter low-power states on a per-chip-select basis.

Once a request is queued to the controller for that DRAM, the hardware will automatically wake from the low-power state if it entered that state via hardware control. This can take a relatively long amount of time since the controller may be required to issue a number of refresh cycles first. For software-initiated self-refresh, software must issue a register write to start the transition from self-refresh back to a fully-powered state.

Deep power-down must be explicitly requested by software, since it causes DRAM data to be lost. Software must disable new incoming requests to the MSS, wait for all outstanding in-flight requests to retire, and then write a register to transition to the deep power-down state. To exit the state, software must reinitialize the DRAM and controller before re-enabling transfers. Deep power-down mode may be enabled for each chip-select in a Protocol Arbiter independently and each Protocol Arbiter independently.

Normal active bank power optimization is managed principally by the Protocol Arbiter. The arbiter will arrange to close banks as soon as possible and to hold requests for as long as the latency timer will allow, with the intent of grouping same-page accesses into a coherent burst.

18.4.2.2 Request Dribble

The “OSIdle” use cases (with or without display active) are important cases of active power management. Display and audio clients can be prone to “dribbling”, that is, sending isolated requests at infrequent intervals.

There are two main causes of dribbling:

- Isochronous dribble, caused by a low periodic request rate. This can be (and should be) managed in the Display client by using hysteresis (high and low watermarks) on the request buffer to group requests into bursts.
- Clock crossing dribble, which is caused by a large disparity between the client clock and the Arbitration Domain clock. If the client’s clock is slow (which it often is for Display), individual requests will have several idle cycles between each request. In general, this is unavoidable by the client.

It is difficult for the Protocol Arbiter to handle the second type of dribble when it is under load, because it cannot “see” a large enough window of requests to group the traffic efficiently. Low-data-rate isochronous clients should have client-side request hysteresis to handle isochronous dribble. The Protocol Arbiter can handle clock-crossing dribble when it is under load, because the delays are smaller and it will schedule other traffic preferentially while waiting for the dribbling traffic to accumulate. When in the OSIdle use case, the Protocol Arbiter will not have enough traffic to keep the client dribble from becoming DRAM dribble (isolated requests that open and close the DRAM pages more often than necessary).

To manage DRAM dribble, the controller adds the concept of the “*hysteresis hold-off*” (also known as iso holdoff), which is a per-controller state bit. When the hold-off is active, the controller is held off from issuing activate commands to open new DRAM pages. Incoming requests will back up in the Protocol Arbiter’s row sorter and store buffer until the hold-off is released.

The hold-off is enabled whenever an individual device goes idle due to lack of work. It is released automatically whenever a latency timer expires, or the Protocol Arbiter stalls due to fullness. The net effect of the hold-off is to group *all* requests for the DRAM into activity bursts.

The hold-off causes a delay in requests transitioning from a memory-idle state. This can affect some traffic in a negative manner – principally requests that are low-latency like the CPU. To keep the hold-off from impacting these requests, software has a per-module configuration bit that disables the hold-off for certain known-latency-sensitive clients. Whenever a request with the holdoff-disable bit set enters the Protocol Arbiter, the hold-off is released. The holdoff is enabled whenever an individual device/rank goes idle due to lack of pending requests. The holdoff is released automatically whenever a request expires, the Protocol Arbiter stalls due to fullness, a request from a non-hysteresis client is received, or the bank-queue length reaches a programmable threshold. The net effect of the holdoff is to group requests for the DRAM into activity bursts.

18.4.2.3 Clock Frequency Scaling

A final aspect of power management is scaling the Memory System clocks. In principle, there are two sets of clocks that can be controlled:

- The Memory System “arbitration clock”, which is the clock used by the Arbitration Domains. Changing this clock reduces the overall bandwidth of the system and reduces the power required by the client interfaces and cross-chip Memory System routing.
- Individual Protocol Arbiter clocks. These reduce the bandwidth available to a particular memory pool and reduce the power consumed by the external pins and the external DRAM. Some Protocol Arbiters may be synchronous to the arbitration clock and not be explicitly controllable via a separate clock enable.

Tegra X1 devices have a single clock domain for the Arbitration Domain and the Protocol Arbiter: mcclk. The command controller runs on a second clock called emcclk, which matches the frequency of the command bus on the DRAMs. Mcclk can be configured to run at 1x, 1/2x, or 1/4x of emcclk. Mcclk and emcclk are phase-aligned.

Both clocks may be changed as part of frequency scaling. Changing the arbitration clock rate affects the Arbitration Domain’s idea of the latency timeout. Software should change the divide-down register to keep the scale of latency timeout “ticks” to a (relatively) constant time unit. Generally a tick time of 30 ns is used since it divides evenly into most of the common DRAM frequencies, but any value of tick granularity may be chosen.

Both the arbitration clock and memory clock have separate divide-downs.

Changing the Protocol Arbiter clock rate affects the relative memory timing parameters. The DRAM timing parameters are double-buffered to allow software to update the timing values for the next clock speed setting and then simultaneously switch to using the new values by writing a trigger.

The DRAM protocol itself may require further clock-change restrictions. To further facilitate the hardware/software handoff and reduce latency for the clock-change operation, a hardware handshake exists between the clock controller module and the Protocol Arbiter.

The MSS maintains two copies of each configuration register that may need to change due to a clock-change, the “active” and “assembly” copies; this is also known as a “shadowed” register. The active copy is what the MSS is configured to use, and cannot be written to. The assembly copy can be written to at any time, and it can be copied onto the active via a trigger mechanism. The clock-change handshake includes a hardware-initiated trigger for this shadow update.

To assist in full-system power analysis, the Protocol Arbiter should include enough statistics about the DRAM bus cycles to be able to calculate the DRAM power consumed.

18.4.3 Device Management

The EMC is primarily responsible for DRAM device management (ZQCAL, thermal management, etc.).

The exception to this is that the MC schedules refresh commands to the DRAM.

18.4.4 Translation Management

Surface allocation for most Tegra X1 clients is straightforward: software allocates the surface out of the device’s virtual space, and then backs the mappings with physical pages. The mapping between virtual and physical addresses is described using PTE and PDEs in the SMMU’s page table.

GPU surfaces have their own virtual address space that is translated by the GMMU within the GPU. The GMMU either translates requests into physical addresses that bypass the SMMU or into a second virtual address space that is then translated into physical addresses by the SMMU.

18.4.5 Statistics and Debugging

Statistics are used for performance monitoring and for generating usage information for guiding dynamic voltage and frequency scaling. Hardware will provide a set of statistics counters that are used for DVFS, both for global MSS bandwidth and for individual PA bandwidth. A separate set of counters with additional filtering support will be available for performance monitoring and debugging.

The hardware blocks illegal transfers to memory. Some transfers are based on physical DRAM limitations (addressing outside of the address map), permissions (addressing “secure” physical memory from a non-secure client) and from the SMMU (accessing an unmapped page). All of these are reported via a fault interface which records the transfer information of the last faulting address, including the following data:

- Virtual address of fault
- Module and client ID of fault
- Fault type

Faulting transfers continue through the system, but faulting reads return all-ones and faulting writes discard the write data. Writing a fault triggers a (maskable) interrupt.

18.4.6 Coherency and Ordering

For coherence and ordering:

1. The MSS does not guarantee the coherency between different clients if the write-response tag has not been provided back to the clients.
2. The MSS guarantees that if a write is sent with a write-response tag and that tag is returned to the client, no other read or write issued after the response is received from anywhere in the system can pass that write.
3. The MSS guarantees that within a single client, write responses will complete in-order.
4. For PCIe ordered clients, the MSS guarantees that all writes will complete in-order.

For the most part, the memory system is fire-and-forget. Software however needs to be involved in the configuration, translation, and power management services as described above.

18.5 Software Interfaces

The primary software interface for configuration is via the APB bus registers.

- Initial configuration
- Power management
- Device/rank management
- Translation management
- ISO client bandwidth allocation (per use case)
- Statistics and debugging

18.5.1 Boot

Tegra X1 devices have a two-stage boot:

- The Boot ROM is responsible for getting the chip out of reset to where driver-level software can take over.

- The Boot Loader or the OS-recovery code is the portion of the driver responsible for completing the boot process to the general-purpose OS.

The Boot ROM is an actual ROM built into the chip, and the Boot Loader and OS recovery section are externally loaded code. The Boot Loader is only used for Cold boot; Warm boot utilizes operating-system recovery code.

Tegra X1 devices can “boot” in two ways: “Warm boot” or “Cold boot”. Cold boot is defined as booting from a fully powered-off or fully reset state. Warm boot is defined as recovery from the lowest-power state (LP0), where the external DRAM has been held in self-refresh mode while the Tegra X1 core was powered off. Warm boot use case occurs much more often; for example, it is the use case hit whenever a SmartPhone wakes from Standby to answer a call.

18.5.1.1 IRAM Use

Tegra X1 device contain a protocol arbiter for accessing IRAM (ARC). This path is used only during boot.

The use of this path does not require the SMMU to be set as the IRAM addresses are expected to be physical addresses. The MC redirects requests to IRAM before the SMMU translation.

This is an additional step in the boot process for these scenarios. ARC_CLK_OVR_ON must be enabled to use the ARC path. Once all IRAM accesses are done for boot, ARC_CLK_OVR_ON should be disabled.

18.5.1.2 Cold Boot

Before Memory Subsystem boot can begin, the Boot ROM must read the Boot Configuration Tables (BCTs) from flash memory (slow storage) into on-chip RAM. These tables include the configuration data for up to 4 initial DRAM configurations supported. The Boot ROM then reads the strap signals assigned to the SDRAM and begins to bring up the memory subsystem.

The Cold boot Boot ROM sequence:

1. Fetch the BCT values from the storage device and move them to IRAM using AHB redirection.
2. Configure the PLL used by the MC/EMC based on BCT values
3. Wait for PLLs to lock
4. Program necessary always-on static pad configurations in the PMC
5. Configure memory clocks based on BCT values
6. Enable the MC/EMC clocks
7. Release memory subsystem resets
8. Program other necessary pad/pad-macro configuration
9. Program initial configuration for the MC/EMC based on BCT values
10. Perform the DRAM initialization sequence. This sequence is specific to the DRAM protocol used and varies depending on the requirements of a particular device. An example sequence goes as follows, but this may vary with DRAM type:
    a. Apply power to the device
    b. Wait until voltage is stable
    c. Assert DRAM reset
    d. Wait 200 µs
    e. Deassert DRAM reset
    f. Apply stable clocks
    g. Wait 500 µs
    h. Assert and hold CKE high
    i. Precharge all banks
    j. Send 2 Auto-refresh commands

k. Write to Mode register 0, 1, 2, 3

l. Issue ZQ Calibration command

m. Wait for ZQ calibration and DRAM DLL lock time to complete

11. Enable power-saving features such as clock stop, active power down, dynamic-self-refresh
12. Start periodic sequences such as auto refresh, ZQ calibration, auto-calibration
13. Release the memory-initialization hardware lock[1]

After this, the external DRAM and Memory Subsystem are ready for use.

18.5.1.3 Warm Boot

The Warm Boot sequence is very similar to the Cold Boot sequence; many of the same steps are performed. One difference is that instead of powering on the DRAM, it is awakened out of SelfRefresh. Another difference is the configuration data for Boot ROM portion of MC/EMC initialization is stored in the Power Management Controller (PMC) Scratch registers.

The PMC Scratch registers are in the Always-On partition, and are powered during LP0. The power and area limitations of this implementation restricts the number of PMC Scratch registers available; there are a smaller number of PMC Scratch Registers than the number of bits needed to fully configure the MC and EMC. The Scratch Registers are only used to store the minimum required configuration to get to working DRAM. The Boot ROM may not be able to Warm boot to full speed using only the Scratch Register data; if this limitation is confirmed, the Boot Loader may be required to mimic DVFS software and reprogram the MC/EMC to full-speed operation using data stored in DRAM. It is up to the LP0 Entry software code to ensure the proper configuration gets written to the Scratch Registers.

The Warm boot code provided by the MSS includes algorithms for deriving the approximate MC arbiter performance parameters from the EMC timing parameters. The EMC has provided generation code for the packing/unpacking calls for the scratch registers to ease the software maintenance burden.

The Warm boot Boot ROM sequence:

1. Configure memory PLLs and memory clocks based on Scratch values
2. Wait for PLLs to lock
3. Configure memory clocks based on scratch values
4. Enable the MC/EMC clocks
5. Release memory subsystem resets. Program any necessary static pad configuration that was powered-off in LP0
6. Program any necessary static pad configuration that was powered-off in LP0.
7. Program initial configuration for the MC/EMC based on Scratch values
8. Release the DRAM from software-controlled Self Refresh. This sequence is specific to the DRAM protocol used and may further vary depending on the requirements of a particular device. An example sequence goes as follows:
   a. Apply stable clocks
   b. Assert and hold CKE high
   c. Issue ZQ Calibration command
   d. Wait for ZQ calibration and DRAM DLL lock time to complete
   e. Write to Mode registers if different from LP0 entry values
   f. Issue refreshes to ensure tREFI is satisfied
9. Enable power saving features such as clock stop, active power down, dynamic-self-refresh
10. Start periodic sequences such as auto refresh, ZQ calibration, auto-calibration
11. Release the memory-initialization hardware lock

1.AHB_ARBITRATION_XBAR_CTRL.MEM_INIT_DONE

After the Boot ROM sequence is complete, the operating system restore code may access DRAM and restore a different or higher-speed operation. The operating system may also restore translation or client configuration settings before allowing any user code to execute.

18.5.2 Security Apertures

To secure the MC registers, the CPU complex's page table entries for the MC register space must be marked as secured. Then to access the registers securely, the CPU must be switched into secure mode. For more details about these CPU operations, please read the appropriate ARM Architecture Reference Manual.

To secure a region of EMEM, the page table entries for that region must be marked as secured. Then the MC registers must be configured for the secured region, using the CPU in the secured mode:

- RegWrite(MC_SECURITY_CFG0, security_aperture_base_address);
- RegWrite(MC_SECURITY_CFG1, security_aperture_size_in_megabytes);

After the registers are updated, the CPU must be in secure mode to access the described region of memory:

- Video Protection region
- Security Coprocessor Secure region

For more information on handling security violations, see Section 18.5.12.2: MC Interrupts below.

18.5.3 Software Client-Groupings

The Memory Subsystem inherently implements a hardware mapping of memory clients ("MCCIFs") to super-clients, and super-clients to the hardware modules. Additionally, the MSS implements a system of mapping modules to "software groups" (also known as SWGROUP or SWNAME). This allows a single point of control for all clients for that hardware "engine". Often the list of clients in a SWGROUP is the same as the list of clients in the module that implements the hardware engine.

Each SWGROUP has registers associated with it that select an ASID (Address Space Identifier) and determine whether SMMU translation is enabled or disabled for the client. Normally, each client is associated with one SWGROUP, thus has one ASID associated with it. A few clients have multiple SWGROUPs associated with them, which are selectable on a transaction-by-transaction basis via a 1- or 2-bit SWID flag provided by the client.

The following table lists the encoding of SWID_0 and SWID_1, when the SMMU is enabled. If the SMMU is disabled, SWID has no meaning.

SWID_1	SWID_0	PPCS	Comments
0	0	PPCS	Access uses PPCS address space for translation at SMMU.
0	1	PPCS1	Access uses PPCS1 address space for translation at SMMU.
1	0	PPCS2	Access uses PPCS2 address space for translation at SMMU.

The mapping of SWGROUP to modules for Tegra X1 devices is listed in the following table.

SWGROUP	Modules	Translatable?	Further Description
AFI	PCIe	Yes	PCIExpress
AVPC	AVP Cache	Yes	ARM7 Audio-Video Processor (AVP)
DC	Display head 0	Yes(1)	Display reads, head 0, with dis2mc_swid=1'b0
DC1	Display head 0, secure client	Yes(1)	Display reads, head 0, with dis2mc_swid=1'b1 (TrustZone®-Display window)
DCB	Display head 1	Yes	Display reads, head 1 (dis2mc_swid=don't care)
GPU	GPU	No(2)	Kepler GPU with a[34]==0

SWGROUP	Modules	Translatable?	Further Description
GPUB	GPU	Yes(2)	Kepler GPU with a[34]==1
HC	Host1x	Yes	Host interface
HDA	HDA	Yes	High-Definition Audio engine
ISP2	ISP2	Yes	Image Signal Processor
ISP2B	ISP2	Yes	Image Signal Processor (second instance)
MPCORE	AXICIF	No	CPU Complex
MPCORELP	AXICIF_LP	No	Shadow CPU Complex (Tegra X1 32-bit devices only)
NVENC	NVENC	Yes	Multi Standard Video Encoder
PPCS	AHBDMA, AHBSLV	No(3)	Clients in the AHB cluster, with ppcs2mc_swid = 2'b00
PPCS1	AHBDMA, AHBSLV	Yes	Same as PPCS, with ppcs2mc_swid = 2'b01
PPCS2	AHBDMA, AHBSLV	Yes(1)	Same as PPCS, with ppcs2mc_swid = 2'b10
PTC	MC (SMMU)	N/A	Misses from SMMU PTC
SATA	SATA	Yes	Serial ATA
SDMMC1A	SDMMC1	Yes	SDMMC controller. Each SDMMC controller has a unique use case and can be run via standard (Windows/Android) or NVIDIA drivers, hence each controller has its own (hardcoded) SWID.
SDMMC2A	SDMMC2	Yes	
SDMMC3A	SDMMC3	Yes	
SDMMC4A	SDMMC4	Yes	
TSEC	TSEC	Yes	Tegra Security coprocessor
NVDEC	NVDEC	Yes	Video Decode Engine
NVJPEG	NVJPEG	Yes	JPEG Engine
VI	VI	Yes	Video Input engine for CSI, VIP
VIC	VIC	Yes	Video Compositor
XUSB_HOST	USB3 Host	Yes	Although the same driver controls both units, each has its own SWNAME to maintain compatibility with the Tegra device.
XUSB_DEV	USB3 device	Yes	

1. The MC defines the ASID mapping on a per software-ID (SWGROUP) basis. To support secure ASID for the SE engine (in the AHB cluster) and TrustZone-Display, additional SWID signals are added to the interfaces. The additional SWID signal allows the MC to map the client to different SWIDs, providing flexibility to use a secure ASID for the SMMU.
2. Although the GPU has one physical read client and one physical write client, it can specify one of two SWGROUPs (GPU and GPUB) for a given request, depending on the value of address bit a[34]. LTCX maps a[34] to the MCCIF SWID field. SMMU translation is disabled for SWGROUP GPU and enabled for GPUB. This is how the GPU indicates whether SMMU translation is needed. The MC also performs special video protection region checks for the GPUB.
3. To support the CPU's view of 16GB memory, part of the 0-2G memory address space is used by the CPU to address DRAM. The devices which were previously making requests in the 0-2G address range, assuming that they would be always mapped to other memory space by SMMU, now overlap with devices making physical address requests in the 0-2G range. An additional bit ppcs2mc_swid indicates which requests need to remain untranslated and which requests need SMMU translation. This additional bit selects between the two SWIDs (for AHB clients). These two SWIDs map to two separate ASIDs, where one indicates SMMU translation and the other indicates no translation.

18.5.4 Virtual Addressing: Using the SMMU

For a discussion of the virtual-address translation features provided by the SMMU, see Section 18.5.4: Virtual Addressing: Using the SMMU.

In Tegra X1 devices, the CPU has its own MMU and issues physical addresses to the MSS. Thus it is not possible to enable SMMU translation for the CPU.

18.5.4.1 SMMU Initialization

1. For each Virtual Address Space (VAS) to be used, set up the page table in the DRAM. See Section 18.5.4: Virtual Addressing: Using the SMMU for details on the page-table formats supported.

2. For each ASID, set up the pointer to the page-table base and the ASID-wide protection bits. Since the PTB pointers are stored in a RAM, this uses an indirect access mechanism involving two register writes:
   a. Write MC_SMMU_PTB_ASID_0 with the ASID to be modified.
   b. Write MC_SMMU_PTB_DATA_0 with the information for this VAS:
   - Page Table Base pointer
   - Whether non-secure accesses are allowed
   - Whether reads are allowed
   - Whether writes are allowed
3. Assign hardware engines (SWNAMES) to the ASIDs and turn on translation for those engines. This involves writes to various MC_SMMU_<SWNAME>_ASID_0 registers.
4. If needed for Hardware Diagnostics purposes, disable or enable translation within a SWNAME on a per-client basis. This involves writes to various bits in MC_SMMU_TRANSLATION_ENABLE_*_0 registers.
5. If desired, secure the ASIDs, which prevents untrusted code from modifying the ASID PTB pointers by enabling a requirement for TrustZone security on register writes that modify the PTBs. This operation involves a write to MC_SMMU_ASID_SECURITY_0. Note that this register itself is secured by TrustZone to prevent it from being modified by untrusted sources, and in some situations writes to it must also be TrustZone secured for them to take effect. In particular, you cannot change a module's ASID with a non-secure write if you are trying to change it to a secure ASID, or if it is already a secure ASID.
6. If desired, enable ASID promotion, which allows clients without security access to inherit the security status based on the page-table translation. This operation involves a write to MC_SMMU_ASID_SECURITY_0.
7. Write PTC_FLUSH_TYPE=ALL to MC_SMMU_PTC_FLUSH_0.
8. Write TLB_FLUSH_VA_MATCH=ALL to MC_SMMU_TLB_FLUSH_0.
9. Configure reserved bits found to be necessary during chip bring-up, if any. There are reserved bits in MC_SMMU_PTC_CONFIG_0 and MC_SMMU_TLB_CONFIG_0.
10. Enable the SMMU by writing SMMU_ENABLE=ENABLE to MC_SMMU_CONFIG_0. Note that this register is secured by TrustZone to prevent it from being modified by untrusted sources, and writes to it must also be TrustZone secured for them to take effect.

The following alternate initialization sequence is used when TrustZone is enabled on Win8, since the UEFI (Boot Loader) is secure, whereas the software memory driver (NVMEM) is not:

1. Hardware initializes the MC_SMMU_TRANSLATION_ENABLE_*_0 register to enable translation. This is done by enabling the bits by default on cold boot (reset) or warm boot (lp0). Note that software depends on this. The software memory driver cannot configure the MC_SMMU_TRANSLATION_ENABLE_*_0 register, since it is TrustZone protected.
2. The UEFI sets up the display for SMMU translation: SMMU_ASID_DC = x, SMMU_ASID_DCB = x.
3. Enable the SMMU: SMMU_ENABLE=ENABLE to MC_SMMU_CONFIG_0.
4. Later the software memory driver sets up the rest of the ASIDs: SMMU_ASID_xxx = y (y is a non-secure ASID).

18.5.4.2 Page Table Modifications after Initialization

Modifications may be made to the page tables while the system is operating, however, care must be taken to avoid modifying an entry that is in active use. Modifying an entry in active use will result in undefined operation; the most likely failure is the use of an out-of-date cached translation. The general rules for safe modification are:

- If no client is using an ASID, then that ASID and its page table can be modified.
- If no client is using any PTE in a PDE, that PDE can be modified.
- If no client is using a PTE, that PTE can be modified.

It is up to software to ensure these rules are followed when virtual address translations need changed. Once software has updated the page-table or ASID mappings, the final step is to perform PTC and TLB invalidates on the modified entries. Note that the granularity of invalidation is at the PTE group, which is the number of PTEs that fit in a memory atom. Software can write:

- MC_SMMU_TLB_FLUSH_0 to flush a specific page group, 4 MB section, entire ASID or the entire TLB.
- MC_SMMU_PTC_FLUSH_0 to flush a specific PTE or PDE from the PTC

18.5.4.3 Collecting Statistics on SMMU Performance

The TLB and PTC each have hit and miss statistics counters. To enable statistics collection, set MC_SMMU_*_CONFIG_0.*_STATS_ENABLE == ENABLE, where * stands for PTC or TLB. The counters will begin collecting information when enabled. To check the results, read the appropriate registers:

- MC_SMMU_STATS_TLB_HIT_COUNT_0
- MC_SMMU_STATS_TLB_MISS_COUNT_0
- MC_SMMU_STATS_PTC_HIT_COUNT_0
- MC_SMMU_STATS_PTC_MISS_COUNT_0

These counters do not saturate when full; instead they wrap from MAX_UINT to 0. There is no explicit reset for these counters; the only reset-like option is to force the counter bits to all 1's using the MC_SMMU_*_CONFIG_0.*_STATS_TEST trigger bits. This is essentially like resetting the counters to -1.

18.5.5 Clock Frequency Management

Software programs the emcclk and mcclk frequencies to provide the required bandwidth and latency while minimizing power.

1. The performance of some devices are highly sensitive to MC clock frequency. Some drivers dynamically specify a minimum emcclk frequency that meets their needs. Drivers make these requests on behalf of the CPU, AVP, HDMI, the various USB controllers, GPU, MPE (NVENC), and ACTMON.
2. ISO drivers specify their ISO bandwidth requirements, which imply a minimum emcclk frequency. The ISO manager (part of IMP) ensures that the bandwidth requested is feasible within system parameters.
3. Thermal throttling dynamically specifies the highest MC clock frequency that should be allowed.
4. The central clock code computes the new EMC/MC frequency as: MIN(MAX(frequency_requests_from_1), ISO_MBps_to_MHz(sum(bandwidth_requests_from_2)), frequency_request_from_3)

18.5.6 DVFS Sequences

DVFS (Dynamic Voltage and Frequency Scaling) is a software feature for controlling system power. A key hardware feature for supporting that is changing SDRAM frequency on the fly with minimal impact on system performance. That indicates the following requirements:

- EMC timing parameters and their corresponding MC arbitration timing parameters need to be adjusted to the optimal performance for the new frequency. These parameters can be derived from DRAM datasheet.
- All trimmer values need to be adjusted for the new frequency. These values need to be characterized across PVT (process, voltage, temperature) and frequencies.
- The pad configurations that can be turned off to save power at low frequencies should also change with frequency change, such as pad terminations and DQS pulls.
- All registers involved in frequency change should be shadowed. That includes the reserved register fields that can possible be turned on only in a certain frequency range. Note that any Hardware Diagnostics feature that has power implications should be considered to be shadowed.
- No software interference should be expected from software after software pulls the trigger to change frequency and before the clock change completes: Software runs on the CPU from SDRAM. Once the EMC blocks the requests to

SDRAM, the CPU may be frozen until the EMC finishes frequency change and unblocks the requests. Note that the CPU can also possibly run out of the contents in cache during clock change, so both scenarios need to be considered.

- The delay caused by clock change should not cause functional failures or visible artifacts.

That results in a design as follows:

- The CAR module handshakes with the EMC on clock change. The EMC handshakes with the MC on shadow register latching. The whole process is done by hardware and transparent to software.
- A rough estimation of clock change delay is 2-6 µs for LPDDR3.
- Supports two clock change mode: self-refresh clock change mode and power-down clock change mode. LPDDR3 supports both modes. Clock change power-down mode is recommended for LPDDR3 since it takes a shorter time.
- For flexibility, EMC has a 16-deep FIFO to hold register programming and release them during or immediately after clock change. That enables one to send DRAM command or latch in unshadowed registers. Here are a few examples:
  - Self-refresh clock change mode requires programming EMC_SELF_REF to put DRAM in self-refresh during clock change.
  - LPDDR3 clock change requires writing mode registers after clock change.

This is a summary of the frequency change process:

- Software turns off features like self-refresh or auto-calibration if necessary to minimize their interference with DVFS sequence.
- Software programs MC/EMC shadow registers for the new frequency.
- Software sets up the appropriate threshold registers in the Display and VI/ISP for *_ready_for_latency_event.
- Software programs EMC clock change flow control registers, such as entering self-refresh before clock change and programming SDRAM mode registers after clock change.
- Software programs the MC/EMC clock source register to trigger the clock change.
- The CAR sends clock change request to the EMC (clock divider and/or clock source).
- The EMC waits for the iso_ready_for_latency_event signal to be asserted. If and while this signal is deasserted, the EMC/MC continues normal DRAM operation.
- The EMC stalls all DRAM commands including refreshes.
- The EMC handshakes with the MC to stalls incoming transactions and waits until EMC pipe is flushed
  Within handshaking, EMC may enters power down state if it is in power-down clock change mode.
- The EMC executes pre-clock change sequence such as entering self-refresh and programming mode register to turn off DLL.
- The EMC latches in shadowed registers and requests the MC to latch in shadow registers.
- The EMC acknowledges the CAR to change clock.
- The CAR changes the MC/EMC clock.
- The CAR deasserts the clock change request to the EMC indicating the clock change is done.
- The EMC executes post-clock-change sequence, such as exiting self-refresh and programming SDRAM mode registers.
- The EMC deasserts the clock change acknowledgement to the CAR.
- The EMC unstalls DRAM commands including refresh and resumes incoming MC transactions.
- Software waits 1µs before continuing.

Software programming sequence on the Tegra X1 clock change sequence:

- Before any clock change, usually at boot time

- Keep these register fields in reset values:
  CLKCHANGE_REQ_ENABLE = ENABLED
  CLKCHANGE_SR_ENABLE = DISABLED
- CLKCHANGE_PD_ENABLE:
  ENABLED: power-down clock change mode (for LPDDR3)

- At each clock change:
  - Disable DSR (dynamic-self-refresh) if it is enabled. There are two ways to do this (software can choose which)
    - EMC_DBG.WRITE_MUX = ACTIVE Read-modify-write: EMC_CFG. DYN_SELF_REF = DISABLE
      EMC_DBG.WRITE_MUX = ASSEMBLY
    - Read-modify-write: EMC_CFG. DYN_SELF_REF = DISABLE
      EMC_TIMING_CONTROL.TIMING_UPDATE = 1
      poll for EMC_EMC_STATUS.TIMING_UPDATE_STALLED == 0
  - Program shadow registers.
  - Program STALL_THEN_EXE_BEFORE_CLKCHANGE = 1
    All following EMC register programming will happen after flushing DRAM requests and stalling further DRAM commands, including auto-refreshes. Note that EMC register reads still will not work from this point until the clock change is complete: issuing an EMC register read in this duration will hang the system.
  - (optional) Program unshadowed EMC registers that need to change with clock change
  - (self-refresh clock change mode only) Program SELF_REF to enter self-refresh:
    SELF_REF.DISABLED = ENABLED
    SELF_REF.DEV_SELECTN = depending on DRAM configuration
  - Program STALL_THEN_EXE_AFTER_CLKCHANGE = 1
    All following EMC register programming will happen after clock change but before unblocking DRAM requests
  - (self-refresh clock change mode only) Program SELF_REF to exit self-refresh
    SELF_REF.DISABLED = DISABLED
    SELF_REF.DEV_SELECTN = depending on DRAM configuration
  - Program DRAM mode registers by MRW (LPDDR3) to change CL/WL/TWR etc.
  - Program UNSTALL_RW_AFTER_CLKCHANGE = 1
    All following EMC register programming can happen with on-going traffic after clock change
  - Wait 1 µs to ensure the EMC programming goes through before programming CLK_SOURCE_EMC.
  - Program CLK_SOURCE_EMC to change the clock:
    EMC_2X_CLK_SRC: MC/EMC clock source
    USE_PLLM_UD: use low jitter clock
    MC_EMC_SAME_FREQ: drive EMC versus MC clock ratio at 1:1, otherwise 2:1
    EMC_2X_CLK_DIVISOR: EMC clock divider
  - Wait 1 µs, in case the next clock change starts before the current one completes, which is possible if the software runs out of cache.

18.5.7 ZQ Calibration Sequence

ZQ calibration is a feature of LPDDR3 to calibrate R on (the output driver) across PVT (process, voltage, temperature). The EMC supports periodically sending ZQ calibration commands to both DRAMs, requiring only initial configuration after booting. The EMC also supports one time calibration commands at booting.

For cold booting, the boot ROM does a ZQ-init on LPDDR3, and then enables periodic ZQ-short.

For warm booting, boot ROM does ZQ-long on LPDDR3 and then enables periodic ZQ-short.

Self-refresh exit:

There are three cases:

- Software-controlled self-refresh as part of LP0/LP1. The device may spend an arbitrarily long time in either of these low-power states, thus could suffer significant drift temperature since the last ZQ calibration. Software must perform ZQ-long at self-refresh exit.
- Hardware-controlled self-refresh after extended period of idle. The duration should not be long (if the duration were long, software should have triggered LP1). Normal, timer-initiated ZQ-short should be sufficient.
- Hardware-controlled self-refresh as part of DVFS. The self-refresh duration deliberately is kept small since the DVFS sequence is as short as possible. There is no need to perform a ZQ calibration within the DVFS sequence itself, since DRAM voltage is not affected by DVFS. Normal, timer-initiated ZQ-short should be sufficient.

One time calibration:

- LPDDR3, Write MC_MRW:
  - MRW_DEV_SELECTN: 2 for device 0, 1 for device 1. Or use 0 for both (not used because software must control the delays between commands in boot ROM).
  - MRW_MA: 10
  - MRW_OP: 0xFF for ZQ-init. 0xAB for ZQ-long, 0x56 for ZQ-short

Periodic calibration:

- EMC_ZCAL_INTERVAL: Number of microseconds to wait between periodical ZQ commands. Program this register to 0 to disable periodic ZQ calibration
- EMC_ZCAL_WAIT_CNT: Number of clocks to wait after each ZQ command before other commands
- EMC_ZCAL_MRW_CMD:
  - ZQ_MRW_DEV_SELECTN
    2 for device 0 only, 1 for device 1 only, 0 for both devices
  - ZQ_MRW_MA
    LPDDR3: 10
  - ZQ_MRW_OP
    LPDDR3: 0x56 for ZQ-short; 0xAB for ZQ-long

18.5.8 MRR Sequence

Mode Register Read (MRR), if supported by DRAM, can be used for reading the manufacturer ID to identify DRAM vendors, or reading back from the temperature sensor, if applicable to DRAM type.

Simple sequence:

- Before any MRR, such as at booting:
  - Set MRR_BYTESEL and MRR_BYTESEL_X16 corresponding to data byte sizzling, if applied on board. The reason is that LPDDR3 always returns MRR at low byte of a data strobe, which is the only case that EMC needs to know about swizzling.
- For each MRR
  - Check EMC_STATUS.MRR_DIVLD = 0. If not true, read EMC_MRR until it becomes true
  - Issue MRR:
    MRR_MA: the index of the mode register to read back
    MRR_DEV_SELECTN: 2 for device 0, 1 for device 1. 0 or 3 are illegal. If both devices are desired, repeat this sequence on each device.
  - Poll for EMC_STATUS.MRR_DIVLD = 1
  - Read EMC_MRR. Data is in the MRR.DATA field

18.5.9 Auto-Calibration Sequence

Auto-calibration is for periodically calibrating the output driver of Tegra X1 pads. The EMC supports periodic auto-calibration without software interference, usually initiated at booting.

To enable auto-calibration:

- Program AUTO_CAL_INTERVAL: the number of microseconds between two auto-calibrations
- Program EMC_AUTO_CAL_CONFIG:
  AUTO_CAL_START = 1
  AUTO_CAL_ENABLE = 1
  AUTO_CAL_OVERRIDE: 1 to enable override, 0 to use normal auto-calibration
  AUTO_CAL_PD_OFFSET: override value
  AUTO_CAL_PU_OFFSET: override value
  AUTO_CAL_E_CAL_UPDATE: number of EMC clocks E_CAL_UPDATE is asserted
  AUTO_CAL_STEP: number of microseconds between each calibration step

To disable auto-calibration:

- Program AUTO_CAL_INTERVAL to 0

To re-enable auto-calibration after disabling it:

- Program AUTO_CAL_INTERVAL to the new value

18.5.10 LP0/LP1 Entry/Exit

LP0 and LP1 are both power saving modes with DRAM in self-refresh mode. The only difference is that the MC and the EMC are ON during LP1 but powered OFF during LP0.

LP0 and LP1 entry share the same sequence:

- Disable DSR (Dynamic self-refresh) by read-modify-write EMC_CFG. DYN_SELF_REF = DISABLED then program EMC_TIMING_CONTROL.TIMING_UPDATE = 1 to latch in the shadowed register EMC_CFG
- Wait 5 µs for worst case DSR exit time
- Program EMC_REQ_CTRL to stall all transactions

  STALL_ALL_WRITES = 1

  STALL_ALL_READS = 1
- Wait until the EMC pipe is flushed
  poll EMC_EMC_STATUS.NO_OUTSTANDING_TRANSACTIONS == 1
- Enter self-refresh

  EMC_SELF_REF.SELF_REF_CMD = 1

  EMC_SELF_REF.SREF_DEV_SELECTN: 2 for device/rank 0 only, 1 for device/rank 1 only, 0 for both
- (optional) Wait until the device/rank is in self-refresh

  poll EMC_EMC_STATUS.DRAM_IN_SELF_REFRESH ==

  1: dev0/rank0 is in self-refresh; 2: dev1/rank1 is in self-refresh; 3: both devices/ranks are in self-refresh

LP0-exit is also called warm boot. Refer to Section 18.5.1.3: Warm Boot

LP1-exit:

- Exit self-refresh

  EMC_SELF_REF.SELF_REF_CMD = 0

  EMC_SELF_REF.SREF_DEV_SELECTN = 2 for device/rank 0 only, 1 for device/rank 1 only, 0 for both
- poll for self-refresh completes

poll for EMC_EMC_STATUS.DRAM_IN_SELF_REFRESH == 0

- Program EMC_REQ_CTRL to unstall all transactions

  STALL_ALL_WRITES = 0

  STALL_ALL_READS = 0

- (optional) if it is desired to re-enable DSR (possibly turned off by LP1 entry sequence): Read-modify-write EMC_CFG.

  DYN_SELF_REF = ENABLED

  EMC_TIMING_CONTROL.TIMING_UPDATE = 1 to latch in shadowed EMC_CFG

18.5.11 Performance Tuning

Note: *The capitalization in the sections below refers to the actual register name of the associated setting.*

18.5.11.1 Per Client Knobs

Each client of the MSS has some configuration knobs that affect the behavior of the request in the PA. These knobs are:

- ISOchronous
- HYSTeresis

Most clients set their ISO and HYST bits based on the behavior profile of the client; which is static for all systems. The default programming should be acceptable for these bits.

18.5.12 Errors (Interrupts)

Both the MC and the EMC implement a handful of interrupts to inform the software of completed events or error conditions encountered by the hardware. Both the MC and the EMC have implemented the same style of interrupt interface, so this section will use the MC as the example.

The interrupt interface for MC consists of two registers and a hardware output wire. The output signal is wired to the central interrupt handling module on Tegra X1 devices; from there it can be routed to any of the CPU complexes. The first register, INTSTATUS, contains the interrupt vector for MC. Each bit in this register is set when the hardware detects an interrupt. Writing a 1 to the interrupt vector bit will clear the associated interrupt. The second register is INTMASK, each bit in this register corresponds to a vector bit in INTSTATUS. If the MASK bit for an interrupt is *clear (MASKED)*, the corresponding interrupt will *not* forward the interrupt to the central interrupt handling module. If any UNMASKED interrupt vector bits are set, the MC will assert the interrupt to the central interrupt handling module.

18.5.12.1 EMC Interrupts

The EMC generates interrupts on the following conditions:

- DLL alarm: From the EMC Digital DLL when the output delay code reaches the maximum value.
- Attempt to issue a command to a device that is in self-refresh.
- Read data from the MRR is available (to prevent software from polling for data)
- DRAM clock change sequence complete
- Refresh overflow

18.5.12.2 MC Interrupts

The MC contains two classes of interrupts: address decode errors and performance warnings.

When an address decode error occurs, the offending address is captured in the MC_ERR_ADDR register, and information about the error is captured in one or more MC_ERR_*STATUS registers. The captured information assists developers in debugging the error.

A single request can trigger multiple errors. There are multiple error status registers (MC_ERR_*STATUS) that capture the status of different types of errors, as listed below:

- MC_ERR_STATUS – Multiple violations. When more than one of the following violations occurs, the highest-priority violation is reported (listed below in descending priority order):
  - ERR_TYPE = SECURITY_TRUSTZONE – TrustZone carveout violations
  - ERR_TYPE = INVALID_SMMU_PAGE – Any SMMU translation violation

    Decode error on PDE/PTE (reserved bits non-zero)

    Request violates RW, nonsecure PTB/PDE/PTE bit values
  - ERR_TYPE = DECERR_EMEM - DRAM minimum/maximum allowed memory addresses
- MC_ERR_SEC_STATUS - All SEC carveout violations
- MC_ERR_MTS_STATUS - All microcode carveout violations
- MC_ERR_VPR_STATUS - All VPR carveout violations

The capture registers record the following information about the error:

- the Virtual or Physical address of the error (depending on the type of fault)
- which type of fault the captured error corresponds to
- a read/write indicator
- the requester client ID
- the swap bit sent by the client
- if the error was an INVALID_SMMU_PAGE, then information about the page's protection status is also captured
- (only in MC_ERR_STATUS):
  - whether the page was marked non secure in the page-table
  - whether the page was marked readable in the page-table
  - whether the page was marked writeable in the page-table
  - Note that SMMU page protection bits are formed by ANDing the page protection bits from the Page Table Base, PDE, and PTEs from all three stages of the page-walk. The ANDed protection bits and the type of the provoking transaction are both available in the status register.

Subsequent errors (of any type) will not change the status and address registers until the corresponding interrupt is cleared.

To prevent requests with address decode errors from modifying memory or accessing memory they do not have permission to, the MC “squashes” the requests. A write request that is “squashed” has had its byte-enables forced to all-zeros, this prevents the write data from being applied to DRAM. A read request that is “squashed” will have its read-return data forced to all-ones, this protects the data in DRAM from being read by non-secure sources.

There is one performance warning type interrupt: ARBITRATION_EMEM. It fires when the MC detects that a request has been pending in the Row Sorter long enough to hit the DEADLOCK_PREVENTION_SLACK_THRESHOLD. In addition to true performance problems, this interrupt may fire in situations such as clock-change where the EMC back pressures pending traffic for long periods of time. This interrupt helps developers identify and debug performance issues and configuration issues.

SMMU-Translated Requests and Physical Aperture Errors

All Virtual Addresses are translated through the SMMU, then run through the Physical Aperture checks. There are a number of confusing scenarios when debugging these interactions including:

- SMMU fault followed by any physical aperture error of any sort: because the SMMU translation occurs first, the data captured by the ERR_ADR and ERR_STATUS registers will always be the SMMU fault information. Both interrupt bits are set.
- Decode error on a successfully translated (i.e., no SMMU fault indicated) client request: the PA checks will throw a decode error; the data captured includes the Physical Address, not the Virtual Address. The client request is squashed. The decode-error interrupt bit is set.
- Decode error on a Page-Table fetch (PDE or PTE): the page-table fetch will throw a decode error; the data captured includes the Physical Address of the PDE or PTE, and the client ID corresponds to the PTC. In addition, the data returned to the PTC is modified to turn off all access permissions, marking the page INVALID. This forces the client request to also throw a SMMU fault when it is translated. Both the decode-error and SMMU-fault interrupt bits are set. The PDE or PTE are then cached with those modified access permissions, so any subsequent requests to that virtual page are also SMMU fault.
- TrustZone security violation on a translated client request: the PA checks throw a security error, the data captured includes the Physical Address, not the Virtual Address. The security violation interrupt bit is set. The client request is squashed. This behavior allows the Secure Aperture to stand as a “last line of defense” against SMMU page-table modification attacks.
- Protected-region security violation on a translated client request: the PA checks throw a security error. The data captured includes the Physical Address, not the Virtual Address. The security violation interrupt bit is set. However, the client request is squashed.
- Video Protection-region security violation on a translated client request: the PA checks throw a security error. The data captured includes the Physical Address, not the Virtual Address. The security violation interrupt bit is set.

18.6 Functionality

18.6.1 Addressing

The 34-bit physical address space supported by the MC is 16GB. Clients may support larger (e.g. 40-bit) addresses internally, but address MSBs above those connected to the memory controller are assumed to be zero. Clients may use virtual or physical addresses. If a client uses virtual addresses, the addresses are limited to 32 bits (the input width of the SMMU). The SMMU outputs 34-bit physical addresses.

18.6.1.1 Address Map and Aperture

Tegra X1 device’s physical address space includes apertures for DRAM, I/O devices, and the AHB redirection region, as defined in the address map (see Chapter 2: Address Map). The range from 0 to 2GB is designated as the MMIO region. All non-DRAM devices (including the AHB redirection region) map (sparsely) within the MMIO region. Address decoders above the MC decode the address ranges for the MMIO devices and intercept requests for the respective devices before the requests reach the MC.

Requests that reach the MC map to one of the following ranges:

- AHB redirection range between IRAM_BOM and IRAM_TOM
- External memory (DRAM) between EMEM_BOM and EMEM_TOM (maximum of 8 GB)
- Not Assigned range (if the address does not fall in either of the two ranges above)

The AHB redirection path allows MC clients that need to access boot media to access IRAM during boot. See Section 18.6.7: AHB Redirection Region for details on the use and programming of the AHB redirection feature. Note that the AHB redirection range might not map on top of the external memory range, depending on how EMEM_BOM is set.

For accesses that fall within the Not Assigned region, an error is logged, writes are acknowledged and squashed before the WCAM, and reads return all 1s.

Four additional programmable apertures restrict which devices can access defined regions of DRAM:

- TrustZone region

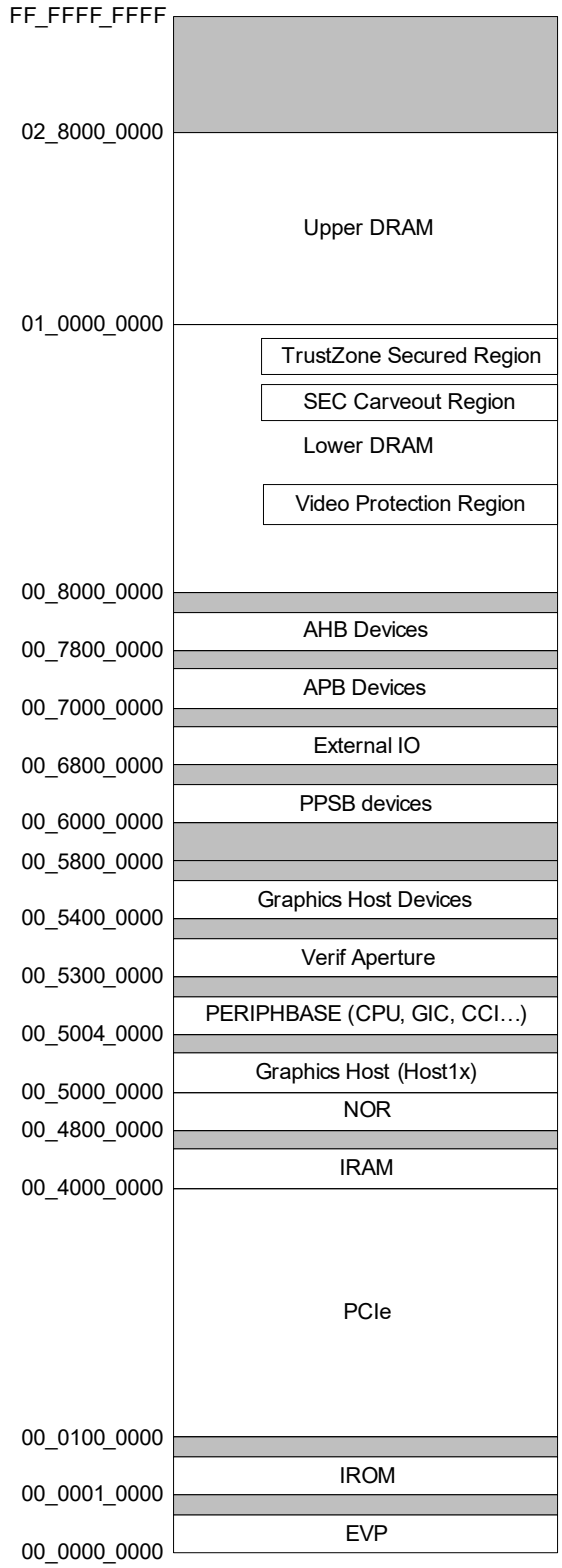
- Video Protection region
- Security Coprocessor Secure region
- Generalized Carveout Protection region

Each of these may be placed anywhere in the external memory range and have programmable size. For details on these programmable apertures, see Section 18.6.8: TrustZone® Security Region, Section 18.6.9: Video Protection Memory Region, and Section 18.6.10: Security Coprocessor Secure Region for more information.

In addition, a range of contiguous physical addresses may be designated as “carveout”. Virtual memory pages mapping to carveout are required to maintain physical contiguity. Carveout is a software-defined region with no hardware aperture detection.

The following figure shows a possible address map for a Tegra X1 system with 8GB of physical memory. For systems with 32-bit addressing (e.g., Windows 8 systems), all memory must live below the 16 GB boundary. If 16GB of DRAM is desired, the external memory range must be configured to map from 0 to 16GB. DRAM is then mapped in the holes in the sparsely populated MMIO space, providing nearly 16GB of addressable DRAM memory. This is the “Swiss Cheese” approach.

Figure 35: Physical Address Map Example for a System with 8 GB of DRAM



18.6.1.2 34-Bit Virtual Addressing

Tegra X1 supports up to 8GB of memory and many SKUs are expected to ship with 8GB populated. The Cortex-A57 and Cortex-A53 CPUs are 64-bit architectures and 64-bit operating systems are expected in the X1 timeframe. So applications will be able to access all 8GB. A 16GB (34-bit) virtual address space is considered necessary to support 8GB of physical memory.

CPU, GPU, and GPU-related clients (DISPLAY, VI, ISP2, NVDEC, NVENC, NVJPG, TSEC) all need to be able to access this 34-bit virtual address space. The CPU and GPU need to be able to access all of it simultaneously.

- CPU does this with its own internal page tables. The MC sees only physical addresses from CPU.
- The GPU can also use physical addressing (performing translation to physical with the GMMU), but this is inefficient. For the GPU, good performance requires dual translation by the GMMU (into an intermediate virtual address space) and then by SMMU, which requires that the GPU have a 34-bit virtual address space.

For the GPU and other clients with 34-bit address interfaces, the ASID registers are extended to point to four ASIDs. The SMMU supports 4GB of virtual address space per ASID, so mapping addr[33:32] into ASID[1:0] extends the virtual address space of a client to 16GB.

18.6.1.3 Client Types

The Memory Subsystem supports five types of clients:

- **32-bit clients with virtual addressing.** These clients use the SMMU for page-based and ASID-based address protection and translation
- **32-bit clients with physical addressing.** These clients either support native scatter-gather or require regions of contiguous physical addresses. Because of the 32-bit input address width, these clients can only access the low 16GB of physical address space. These clients still have ASID protection for secure regions.
- **34-bit clients with virtual addressing.** These clients use the SMMU for page-based and ASID-based address protection and translation. Because the SMMU only handles 32-bit input addresses, the upper two address bits are ignored when operating in this mode.
- **34-bit clients with physical addressing.** These clients either support native scatter-gather or require regions of contiguous physical addresses.
- **34-bit CPUs.** These clients have their own memory management unit and CANNOT under any circumstance use the SMMU for address protection or translation

For Tegra X1 devices, the following list indicates the clients that support 34-bit addressing:

- DFD
- Display
- HDA
- GPU
- ISP2
- (Virtual address client, so only 32 bits are relevant)
- PCIe (AFI)
- SATA
- SDMMC
- TSEC (Virtual address client, so only 32 bits are relevant)
- USB3 (XUSB)
- VI (Virtual address client, so only 32 bits are relevant)

18.6.2 Tegra X1 Granularity

LPDDR3 supports burst lengths of 8. LPDDR4 supports burst lengths of 16. The Memory Subsystem only supports BL8 for LPDDR3 and BL16 for LPDDR4. Because of x64 RAM configs, the minimum quantum to transfer is 64B. When using a 32-bit DRAM channel, the EMC will construct two BL8 requests for LPDDR3 and one BL16 for LPDDR4 to DRAM to service the 64B request from the MC.

To match this, MC transactions carry 64B of data payload. The 64B can be contiguous or can consist of two non-contiguous 32B “sectors” that lie within the same 512B “gob”, each mapping to a different 32-bit for LPDDR3 and 16-bit for LPDDR4 DRAM sub-partition (see Section 18.3.9: Memory Sub-Partitions). Byte access remains possible but involves a 64B access: write accesses can use byte enables to write a set of bytes (including a null set), and read accesses always return two full 32B sectors with potentially unneeded data (also known as overfetch). To efficiently operate in DDR3 mode, most engines in Tegra X1 devices use 64 B memory requests.

The granularity of external access is linked to the burst size programmed for DRAM access. Bursts into each 32-bit DRAM channel for LPDDR3 and 16-bit DRAM channel for LPDDR4 always start at a 32B aligned boundary and consist of 8 data beats for LPDDR3 and 16 data beats for LPDDR4. For x32 configurations, the two 32B sectors in an MC request correspond to two transfers of 8 data beats in LPDDR3 and one transfer of 16 data beats for LPDDR4.

DDR3 and LPDDR3 support different features to enable less data from being transferred than what the Burst Size indicates. The resulting minimum data transfer is indicated in the following table.

DRAM Width	LPDDR3 (BL8)	LPDDR4 (BL16)
x32	32B	32B
x64	64B	64B

Tegra X1 devices support:

- BL8, LPDDR3
- BL16 LPDDR4

18.6.3 Virtual Addressing

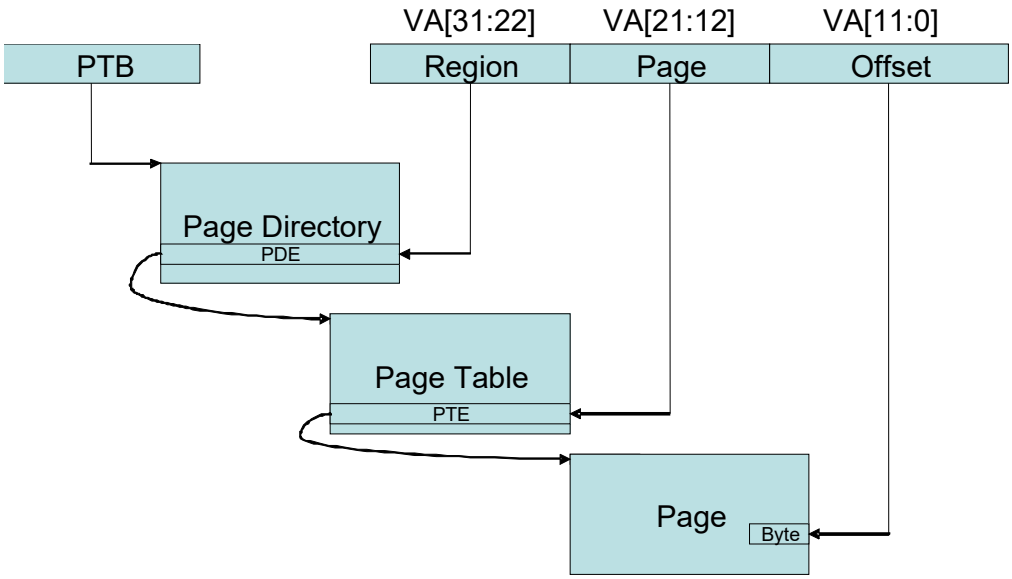
The SMMU is a centralized virtual-to-physical translation for MSS. It maps a 32-bit virtual address to a 34-bit physical address. If the client address is 40 bits then bits 39:32 are ignored. It allows the following:

- Memory allocation for hardware pools (surfaces and buffers) could be made from non-contiguous page-sized chunks of memory
- Areas of the hardware memory map could be marked as protected, preventing access from hardware devices
- Hardware memory access can be controlled on a per-device or per-process basis

The virtual-to-physical translation is via a two-level page table that carves up the virtual address as follows:

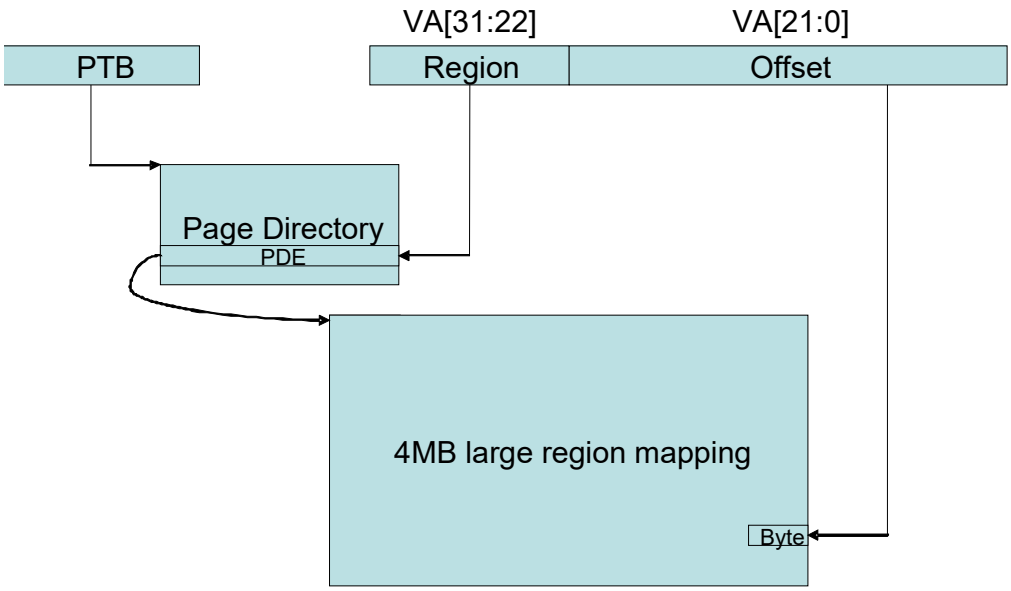
- Bits 31:22 choose a page directory entry (PDE). Each page directory has 1,024 PDEs, each mapping 4MB region.
- Bits 21:12 choose a page table entry (PTE) from a page table. Each page table has 1,024 PDEs, each mapping a 4kB page.
- Bits 11:0 choose a byte within a page

Figure 36: Page Table and Virtual Address



Each PTE or PDE consume 32 bits of memory, so that each page directory or page table consumes one 4kB memory page. In addition, the page directory lookup does an early-out either to map an entire 4MB region of the virtual space as invalid, or to map the region to a contiguous range of physical pages.

Figure 37: Large Region Mapping



The SMMU supports multiple address spaces, each with its own page-table-base register (PTB) and page table. Each address space is tagged with an address space identifier (ASID). This allows for multiple virtual mappings to exist at the same time. Typically, software will assign a different address space to each process that directly uses the hardware drivers, allowing process isolation. The number of ASIDs is 128. Software is responsible for managing the assignments between processes and ASIDs, flushing ASIDs out of the TLB as needed when they are remapped between processes.

ASIDs are assigned based on the SWNAME of the incoming request. An on-chip mapping between SWNAME and the ASID is maintained by the SMMU. For register-based devices, the driver is responsible for updating the mapping between the SWNAME and the ASID when a device in context is swapped between processes.

Untranslated modules are also supported. The SWNAME-to-ASID mapping has a translation-enable bit to allow translation on a per-SWNAME basis. There is also a per-client translation-enable bit for Hardware Diagnostics purposes. This can be used by

any client but is typically used by clients that generate 40-bit addresses. In addition, a global “translation enable” bit disables all translation when cleared (cleared is the default out of reset).If translation is disabled, client address bits 33:0 are sent directly to the output of the TU. If the client only provided a 32-bit address, the physical address bits 33:32 will be set to 0.

If translation is enabled, the lower 32 bits of the client address will be used to generate the physical address. Note that this means that client address bits 39:32 will be ignored if a client sends a 40-bit client address and its associated ASID has translation enabled. This is done to minimize the changes to the SMMU. It is expected that 40-bit clients will typically target ASIDs that have translation disabled.

Both the page and directory tables are stored in DRAM. A translation look-aside buffer (TLB) will be used to cache recently-used translations. Translations that miss in the TLB would incur potentially two additional round-trip DRAM delays, but the latency-scheduling features of the Protocol Arbiter will be used to hide this latency behind the normal latency incurred by the DRAM reordering policy. The TLB is backed by a page-table cache (PTC) to reduce this memory traffic further.

To further improve CPU latency, the Tegra X1 CPU low-latency read paths bypass the SMMU altogether; the hardware also restricts the CPU write paths to untranslated-mode to prevent any configuration mishaps.

18.6.3.1 Page Table Layout

The in-memory layout of a PTE looks like

- 22 bits of physical address (PA[33:12]) for the base of the page
- Protection bits which include
  - A “readable” bit which allows clients to read this page
  - A “writeable” bit which allows clients to write this
  - A “nonsecure” bit which (when unset) restricts access of this page to “secure” TrustZone clients

The remaining bits in the 32-bit word are reserved.

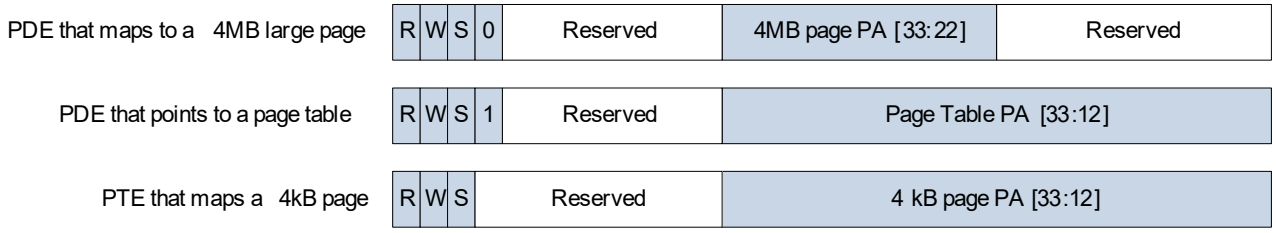
Note: *Wherever physical addresses are used in this document, it is assumed that the device can address 8GB of physical memory.*

The in-memory layout of a PDE looks like:

- A “next level” bit which says if this PDE maps a single 4MB region, or if it points to a page containing a table of PTEs for the second level of translation
- If the “next level” bit is set, there are 22 bits of physical address (PA[33:12]) that point to a page containing the base of the PTE table for this PTE. If the “next level” bit is clear, 12 bits (PA[33:22]) are used to point to the base of an aligned 4MB physical region mapped directly by the PDE.
- The remaining bits set the permissions for the access. For 4MB large pages, these are used directly for the region. For PDEs that point to a second-level page table, these permissions are ANDed with those of the PTE to give the final permission
  - A “readable” bit for the region
  - A “writeable” bit for the region
  - A “nonsecure” bit for the region

The remaining bits in both interpretations are reserved. Reserved bits should be 0, otherwise they can cause an SMMU fault. Because these are compressed in PTC before they are identified as being PDE or PTE, only the subset of reserved bits that are common to all formats (bits [27:22]) will trigger an SMMU fault.

Figure 38: PDE and PTE Formats



The base address of the page directory for a particular ASID is stored on-chip in the ASID table. Changes to this table may require flushing the ASID in question from the TLB.

18.6.3.2 Page Validity and Protection

There is a need on the software side to distinguish between an invalid mapping (which has an invalid PA field) and a mapping that is valid, but has no access. The following pseudo-code describes the recommended conventions.

```
    // page/section/address space permission types
    //
    // note that INVALID implies the PA field is unused.
    // PA is valid for all others, including NO_ACCESS.
    enum Permission {
INVALID        = NvU32(0),
NO_ACCESS      =                              NONSECURE,
SECURE_WRITE_ONLY=     WRITABLE,
WRITE_ONLY     =              WRITABLE | NONSECURE,
SECURE_READ_ONLY= READABLE,
READ_ONLY      = READABLE |           NONSECURE,
SECURE                 = READABLE | WRITABLE,
ANY                    = READABLE | WRITABLE | NONSECURE
     };
```

For more information on handling SMMU faults, see Section 18.5.12.2: MC Interrupts above.

18.6.3.3 Flushes and Page Table Updates

Changes to the page tables require the TLB to be flushed. A software interface for selectively flushing parts of the TLB is provided. The flush command can flush any TLB entries that match a particular subset of virtual addresses. Each address component may be included for matching:

- Match on the ASID, or not
- Match VA[31:22], or not
- Match VA[31:16] or no VA match

Disabling all matches flushes the entire TLB. Note that the TLB is 16 entries wide, so when you match any particular VA, you flush the entire line, which is why the VA matches are down to 16, rather than 12. 4 MB pages are still matched down to VA[16] since they are stored in the TLB using the 4 KB page format.

In addition to the TLB, page table updates require flushing stale entries out of the PTC. This needs to be done on a slot-by-slot basis by writing the PDE or PTE's physical address (PA[33:4]) to the PTC flush command. Each flush flushes 16 aligned PTEs or PDEs (one DRAM atom's worth) out of the cache.

A page table update requires three writes – one to write the PTE to memory, one to flush the stale PTE out of the PTC, and a third to flush the stale VA mapping out of the TLB. However, since a single PTC flush flushes 16 PTEs or PDEs and a single TLB flush can flush the entire TLB, software can reduce the flush traffic by merging flushes from multiple updates.

18.6.4 DRAM Device/Rank□Bank and Page Mapping

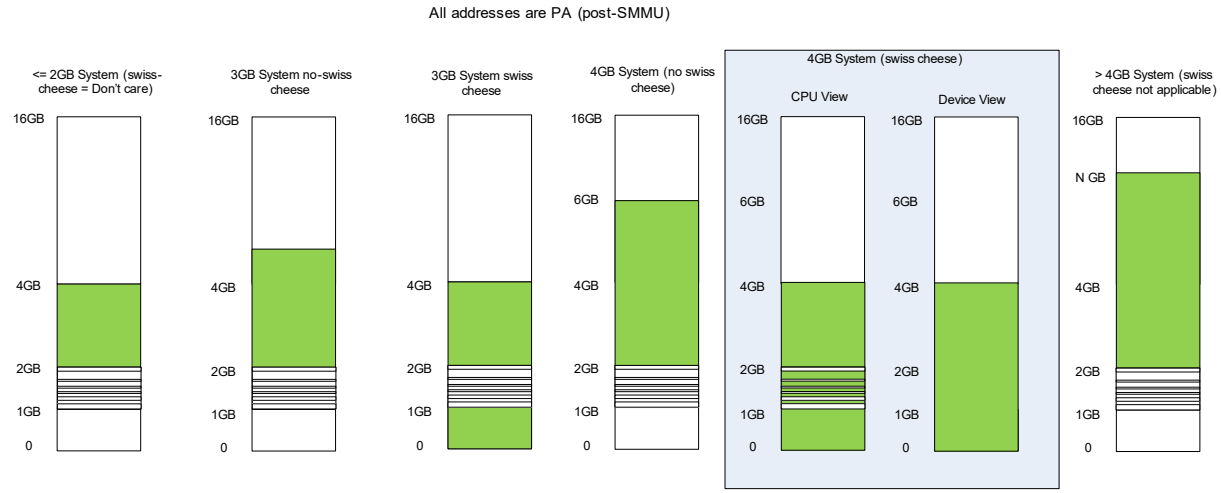
Tegra X1 devices support two modes of addressing; normal addressing mode and swiss-cheese mode. The EMEM_BOM field in the MC_EMEM_CFG_0 register supports the two addressing modes.

EMEM_BOM == 1'b0: EMEM Base is 2GB // Normal AMAP

EMEM_BOM == 1'b1: EMEM Base is 0 // Swiss-cheese mode, EAMAP

The various address ranges for DRAM in both configurations are shown the figure below.

Figure 39: : Physical DRAM Address Space for Configurations



In case of any errors, MC logs the address of the violating request in the MC_ERR_STATUS_0.ERR_ADR_HI and MC_ERR_ADR_0 registers. In case this error address is due to decerr, the logged address is the original Physical Address.

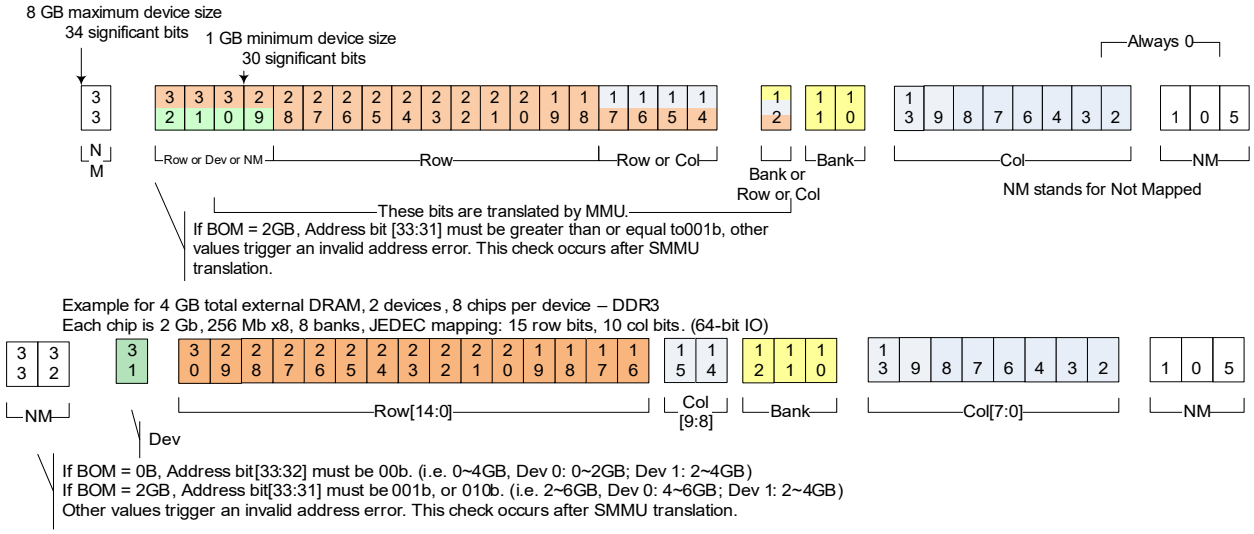
The EMC supports multiple devices with data widths of 8, 16, or 32 bits[1], all identical to form a single rank of either 32 or 64 bits. Up to two ranks are supported.

The bit mapping between the internal linear address and the device/rank, row, column and bank bits is performed as follows:

- The 2 LSBs of the linear address are ignored as the address granularity of the DRAM is 32 bits/4 bytes
- In 64-bit systems, bit 5 of the linear address determines the 32-bit subpartition (with hashing of MSBs)
- In 64-bit systems, bits[9:6,4:2] of the linear address are mapped as column bits [6:0]
- In 32-bit systems, bits[9:2] of the linear address are mapped as column bits [7:0]
- Bits [11:10] are bank bits, possibly hashed.
- Bit [12] is a bank bit if the device has more than two bank bits, possibly hashed.
- The next bits of the linear address are mapped as column bits, as many as remaining after previous mapping
- The next bits of the linear address are mapped as row bits, as many as needed for the selected device
- The next bit is a device/rank bit if device/rank bits are needed

1. A x32 device may be a single x32 chip or two x16 chips in parallel.

Figure 40: Address Mapping Example



The number of sub-partition, bank, column, row, and device bits is limited by the number of address pins available:

- Sub-partition width: 1 or 0
- Bank width: 2 or 3
- Column width: 9 to 12
- Row width: 12 to 16
- Logical Devices/Ranks (also known as chip selects): 1 or 2

When two logical devices/ranks are used, the total memory mapped by the second device/rank must be less than or equal to the first device/rank. The second device/rank also may have a different row, bank, column mapping from the first device.

18.6.5 DRAM Power Modes

The following table maps between the system power modes and DRAM power modes.

	VDD_CORE	VDD_CPU	MC/EMC Power	PLL	MC/EMC Clock	DRAM State
LP0 (deep sleep)	off	off	off	off	off	self-refresh
LP1 (suspend)	on	off	on	off	off	self-refresh
LP2 (idle)	on	off	on	on	on	active

Refer to Chapter 12: Power Management Controller for more details on system power modes.

18.6.6 Bank Hashing

To achieve a good bank interleave, Tegra X1 devices perform a hash of address MSBs into the bank bits.

18.6.7 AHB Redirection Region

During boot, USB3 and flash media (SDMMC/SATA) devices need access to IRAM. Because these clients connect to the MC and do not have a direct path to the IRAM, The MC implements AHB redirection during boot to allow path to IRAM. In this mode, accesses to a programmed memory address aperture are directed to the AHB bus, allowing access to the IRAM.

The AHB aperture is defined by the IRAM_BOM and IRAM_TOM registers, which are initialized to disable this aperture. If the fuse indicates that the boot media is present in one of these clients, that Boot ROM will initialize this aperture to 0x40000000 and 0x4003f000 (1GB, 1GB+IRAM_SIZE), respectively. Once bootup is complete, the boot ROM programs the IRAM_BOM to 0xffffffff and IRAM_TOM to 0x00000000, thus disabling access to IRAM. DRAM is then potentially accessible in this address

range. These aperture register also have an access_control/lock bit. After disabling the aperture, the access_control register should be programmed to lock the registers.

AHB routes the requests targeted to IRAM appropriately and routes all non-DRAM aperture requests (as defined by the swiss-cheese and related configuration) back to the MC as the default DRAM region. Care must be taken to program the IRAM aperture register to prevent a request from being stuck in a loop.

18.6.8 TrustZone® Security Region

The MC can secure a region in the physical address aperture from insecure software sources. The region can be defined on 1MB boundaries in the external memory region, in multiples of 1 MB. If a non-secure request is made to the secured region or if a secure request from the CPU is made outside of the secured region (after this mode is enabled), the MC generates an interrupt (if enabled) and logs the details of the request (address, client ID). After the details are logged, write requests are dropped and read requests are forced to return all 1's, thus protecting the secured region from corruption by the insecure source and avoiding aliasing in the CPU cache.

Only clients configured for security can make secure requests.

After the TrustZone aperture is established, the CPU_STRICT_TZ_APERTURE_CHECK bit in the MC_TZ_SECURITY_CTRL_0 register is set in the MC, which forces correct behavior as follows:

- Secure requests from the CPU can only access TrustZone secure memory space and not access the memory outside TrustZone secure memory space.
- Secure requests from other clients may access the TrustZone secure memory space and also may access memory outside TrustZone secure memory space.
- Non-secure requests from any client can only access memory outside the TrustZone secure memory space.

The register MC_TZ_SECURITY_CTRL_0.CPU_STRICT_TZ_APERTURE_CHECK itself is sticky and cannot be changed once set after cold boot.

The clients that have access to the TrustZone DRAM region are:

- CPU
- PTC
- PPCSCLVMEM (AHB)
- SE
- NVDEC
- DISPLAYT (Trust-Zone Display)
- Host1x

18.6.9 Video Protection Memory Region

This protected memory aperture prevents the CPU (and other unauthorized clients) from accessing data within the region. The MC secures a region in memory from access and modifications by engines that are not part of the video decode and display process.

This aperture is defined by the VIDEO_PROTECT_BOM and VIDEO_PROTECT_SIZE_MB (reset value 0x0) registers. The value 0 for the size register disables this aperture.

In addition to protecting the region in memory, the registers are allowed only secured access by the Boot Loader only. On cold boot when this register is initially programmed, it is also written to the secure PMC-scratch region. On LP0 resume, the register value is restored from this region. This eliminates need for saving this register content on LP0 entry. An additional lock bit locks down the access to this register. On reset the lock bit is cleared. When the lock bit is cleared, the writes are allowed to the VPR aperture registers. When the VPR registers are initialized, the lock bit is set. When the lock bit is set, all writes to the VPR aperture registers are ignored. This lock bit is used for cold boot as well for LP0 exit.

For systems that do not implement VPR, the Boot Loader must still set the lock bit to indicate that programming is completed.

The clients that have access to the VPR aperture are:

- Display
- GPU (GPUB does not have default VPR access)
- TSEC
- VIC
- NVENC
- NVDEC
- HDA

For security reasons, the VIDEO_PROTECT region must be locked by boot loader in devices where the VIDEO_PROTECT security region is available (defined by a fuse). The region must be locked even if the SIZE of the region is zero. Failure to lock this region will prevent the GPU from accessing system memory. The VIDEO_PROTECT region can be locked by writing DISABLE to MC_VIDEO_PROTECT_REG_CTRL_0::VIDEO_PROTECT_WRITE_ACCESS.

Note: *As a part the GPU initialization process, the GPU waits for the VPR and the WPR information from the Memory Controller. VPR and WPR should be in locked state before the Memory Controller can send it to the GPU.*

18.6.10 Security Coprocessor Secure Region

The Security Coprocessor (SEC) engine needs a secured memory space meant exclusively for use by this engine. This memory region, SECR (SEC region) should be accessed only by the SEC engine; no other clients are allowed to read or write to this memory region. This memory region is defined by physical memory range.

The SEC engine (also referred to as TSEC) uses this carveout region as an extension to its I-cache and D-cache. Without this carveout, the TSEC has to encrypt the data before writing to normal DRAM.

The SECR_BOM and SECR_SIZE_MB registers (reset value 0x0) define the 1MB aligned base and size register for this region. In addition to protecting this memory region, these aperture definition registers themselves need to be protected and are allowed to be modified only by the Boot Loader (during cold boot) and the Boot ROM (during LP0 resume) and should be implemented similar to the VPR aperture registers. An additional sticky/secure bit controls this register. On reset this sticky/secure bit is cleared, thus allowing write access to the SECR aperture registers. When the Boot Loader initializes the aperture registers, it also sets this sticky/secure register bit. When this sticky/secure bit is set, all subsequent writes to this register are ignored. On LP0 resume, reset clears this sticky bit; it is set again when the aperture register is restored from the PMC secure scratch space. To avoid saving this aperture register on LP0 entry, the Boot Loader (during cold boot) should store this aperture definition to the PMC secure scratch space.

The size for this carveout region is 1MB.

Any access to this memory aperture by any other client (other than SEC) is ignored (read returns all 1s as the response and writes are dropped) and an error is logged.

18.6.11 Generalized Carveouts

The Tegra X1 Memory Controller has 5 generalized carveout memory regions that can be configured to allow protected access to certain clients: two are reserved for GPU (for Write-Protected-Region), one for NVDEC, one for TSEC0, and one for TSEC1. These clients use generalized carveouts to protect Falcon micro-code from external tampering, but the use of generalized carveouts is not limited to write-protection regions only.

In addition to client ID, generalized carveouts require clients to send two additional fields (via sideband signals): a*perture_id* and *access_level*. Requests to generalized carveouts are then rejected or accepted based on the tuple {*aperture_id*, *access_level*,

client_id, *smmu_translated*} and the configuration of the carveouts. The fields *aperture_id* and *access_level* of clients that do not use generalized carveouts are tied to zero (at the MCCIF).

Fixed-function carveouts supported in Tegra X1 devices:

- TrustZone region
- Video Protection region
- MTS Carveout Protection region

18.7 Memory Tiling

Tegra X1 supports the memory tiling formats described below.

Table 81: Client Shared Tiling Formats

Format	Status	Used for	Description
Pitch Linear	Supported by some clients	Software-compatibility, clients that linearly access memory	Simple array ordering
Block linear 16Bx2 with Tegra X1 sector ordering	Supported by all clients	Kepler color surfaces, clients that access memory in blocks or vertically	Square DRAM page arrangement (blocks) with folded atoms
Legacy (Prior to Tegra 2 devices)	Not Supported		

The parameters associated with each tiling format are listed below:

Table 82: Surface Descriptor Parameters

Parameter	Used by	Description	Values	Minimum Alignment
Layout	All	Surface layout	PITCH, BLOCKLINEAR	N/A
Base Address	Pitch	Starting address of surface, in bytes	32-bit virtual address (40-bit virtual address within the GPU) -or- 34-bit physical address	64B (general) 128B (GPU) 256B (NVENC, VIC)
Pitch	Pitch	Line-to-line stride in bytes	Large enough for at least 4,096 pixels in bytes (some clients may support larger values)	64B (general) 128B (GPU) 256B (NVENC) up to 256B (VIC)[a]
Base Address	Block Linear	Starting address of surface, in bytes	32-bit virtual address (40-bit virtual address within the GPU) -or- 34-bit physical address	512B (one gob)
Block Width	Block Linear	Width of block in gobs	ONE_GOB (this is the only setting supported on Tegra X1)	N/A
Block Height	Block Linear	Height of block in gobs	[ONE_GOB, THIRTYTWO_GOBS] (SIXTEEN_GOBS is the normal setting, but ONE_GOB may be needed for display surfaces outside carveout	N/A
Block Depth	Block Linear	Depth of block in gobs	[ONE_GOB, THIRTYTWO_GOBS] (values greater than ONE_GOB are only supported by the GPU)	N/A
Width	Block Linear	Width of surface in samples	[1, 4096] samples (some clients may support larger values)	1 sample

a. The pitch alignment restrictions for VIC depend on its cache line shape: 256Bx1: 256B, 128Bx2: 128B, otherwise: 64B.

(x,y) to linear address translation is the responsibility of clients. The MCCIF only supports linear address inputs. Contact your NVIDIA representative for access to libraries with translation code to simplify implementation.

18.7.1 Block Linear

The block linear formats map batches of contiguous addresses into rectangular *blocks*, and tile these blocks linearly. Blocking serves two purposes: 1) it maps a DRAM page into a rectangular region, rather than into a one-pixel-tall strip, to capture locality in a graphics geometry stream or other 2D-oriented client, such as an MPEG encoder or decoder, and 2) it makes it possible to predictably interleave banks in x and y, minimizing bank conflicts over a large region.

Blocks themselves are arranged from *GOB*s (*Groups of Bytes*). A Kepler GOB is 512 bytes arranged as 64x8 bytes. GOBs are stacked vertically to form a block. The number of GOBs stacked vertically in a block is controlled by an additional surface parameter called the *block height*. The recommended block height for most buffers in Tegra X1 devices is 16 GOBs (128 lines). This supports 8x8 bank interleaving. For buffers for which linear display access is more important than access by GPU, VIC, or other block-oriented engine, block height can be set to 1 GOB, providing more locality for the display client.

Blocks are arranged horizontally into a *Row of Blocks (ROBs)* to fill out the surface to the width value (or slightly beyond, if the width is not a multiple of the 64B block width). Upper address bit swizzling in the bank swizzling algorithm generally makes it unnecessary to pad the surface width further.

18.7.2 Sub-Partitions

The 64-bit DRAM channel can be divided into two sub-partitions with DQ[31:0] forming sub-partition A and DQ[63:32] forming sub-partition B.

Each 64B atom is divided into 2 sectors of 32B. One sector contains bytes 0 – 31 while the other contains bytes 32 – 63.

18.7.3 Software Guidelines

Software should allocate surfaces respecting the minimum alignments specified in Table 82. This section gives additional alignment restrictions for certain clients and advice on allocating surfaces for best performance.

Certain clients, such as NVENC, NVDEC, NVJPEG, and VIC use 32-bit registers (left-shifted 8) to specify 40-bit base addresses of a buffer. Base addresses for these clients must be aligned to a minimum of 256B.

Certain clients, such as NVENC and VIC, have a fixed cache line size and read all data within a cache line, even if it is beyond the nominal height and width of the surface. For such clients, surface width and height need to be padded to at least the next multiple of the cache line size so the unit does not read beyond the end of the buffer.

Surface Type

- **Block linear** format generally provides best performance and should be chosen by default.
- **Pitch format** may be preferable if producer and consumer clients perform linear accesses and both clients support pitch.
- **Private formats** are used by a few units, such as NVENC, NVDEC, and NVJPEG. For private formats, the client-required allocation and padding rules must be followed.

Block Linear

- **Alignment**. Align surface start address as follows:
  - 512B is the minimum alignment for correct behavior. Performance will be poor (no zero-bandwidth clears, poor page locality).
  - 1KB alignment is better. Zero bandwidth clears will work. Bank interleave will be suboptimal.
  - 8KB alignment is best. All compression features work. Bank interleave assumes 8KB alignment. Use this setting for large buffers.

- **Block height** should normally be programmed to SIXTEEN_GOBS (128 lines). For certain surfaces for which display is the primary client, block heights of ONE_GOB or TWO_GOBS are recommended. Surfaces to be accessed by the GPU texture unit may require a smaller block height under certain conditions.[1]
- **Width** is typically prescribed by the application. Hardware allocates a block linear surface to be an integral number of blocks wide, so its allocated width is width*Bpp rounded up to the next block boundary (64B), which is sufficient padding in almost all circumstances.[2]
- **Height padding**. Surfaces should be an integral number of blocks tall. If a client such as NVENC or VIC has a cache line that is taller than the block height, additional row(s) of blocks may be required for padding.

Pitch

- **Alignment**. Align surface start address to a minimum of:
  - 128 B if surface is to be accessed by the GPU
  - 256 B if surface is to be accessed by NVENC, NVDEC, NVJPEG, or VIC (due to the 8-bit left-shift issue)
  - 64 B otherwise
- **Pitch**. Surface pitch must be a multiple of:
  - 128 B bytes if surface is to accessed by GPU
  - 128 B if surface is to be read by VIC with 128x2 cache line shape
  - 256 B if surface is to be read by VIC with 256Bx1 cache line shape
  - 64 B otherwise.
- **Height padding**. For clients, such as NVENC and VIC, additional lines of padding might be required beyond the nominal height of the surface. For example, when the VIC is reading a source buffer with an internal cache line shape of 32x8, the source buffer must be a multiple of 8 lines tall.

18.8 Latency Allowances Timestamps, Deadlines, and Slacks

The Tegra X1 Memory Controller supports clients with highly differing latency requirements. It achieves this by calculating a deadline for each request as it enters the memory controller and then prioritizing requests that reach their deadline and still have not been serviced.

When all requests are within their deadline, the arbiter grants requests to maximize efficiency (Efficiency Arbitration Mode). If any request(s) have reached their deadline, the arbiter operates in Latency Arbitration Mode, giving high priority to the expired request(s). For a variety of reasons, an expired request may not be able to be granted immediately. Expiration Time is the time between a request reaching its deadline and the time it is actually granted by the arbiter.

18.8.1 Deadline and Slack Computation

When a request is stored in the partition client (PC) FIFO, a deadline is computed for it. The deadline is computed by adding the latency allowance (LA) of the issuing client to the current timestamp. The LA is programmable on a per-client basis. Slack is defined as Deadline – Timestamp with wrapping arithmetic as described below. Negative slack means the request has been pending in the EARB for longer than its latency allowance and should receive latency-mode arbitration.

To decouple the concept of time in the EARB from the clock period of the mcclk, the units of deadlines, LAs, timestamps, and slack are all stored in units of ticks. A tick is the wall-clock granularity; nominally 30ns. There are configuration variables to tell the TIMER unit how to translate mcclk cycles into the appropriate timestamp increments for the current clock-frequency.

When viewing the MC statistics, a positive slack indicates the requested latency allowance was achieved; negative slack indicates the request was pending in the arbiter longer than the programmed latency allowance. Larger (signed) slack values

1. For buffers that are to be accessed using the GPU texture unit, the BlockHeight parameter must match that used by the texture unit. In particular, note that if a texture surface is smaller than the specified size in blocks, texture uses a shrunken blockHeight instead of the nominal parameter specified in the Texture Header.
2. If VIC cacheline shapes of 128x2 or 256x1 are used (generally not recommended for block linear), width must be padded up to a multiple of the cache line width. If this is not possible, a narrower cache line shape must be used.

are indicative of less latency. Since the arbiter represents latency internally in units of deadline timer ticks, the slack value reported in these statistics is also in units of ticks.

18.8.2 Determining the Latency Allowance

NVIDIA determines the latency allowance setting for each client. In many cases, the latency allowance can be statically programmed. In some cases, it must be dynamically programmed based on client bandwidth and/or memory clock frequencies.

Before giving programming guidelines for different client classes, some term definitions are below:

- Latency Tolerance. The latency a client can tolerate while running at its design bandwidth. This may be governed by a) the size of internal latency buffers within the client or b) the size of the client's MCCIF response buffer (which needs to be large enough to store all of the client's requests outstanding in the memory controller). The latency allowance should be less than the Latency Tolerance minus the expected Expiration Time and minus pipeline latencies not accounted for in the latency allowance, to avoid stalling the client.
- Self-queuing time. The time required to process all requests from a client that could reside in the row sorter at once. For example, if a bandwidth-soak client floods the arbiter with requests, such that the row sorter is filled with requests from that client, if requests are serviced with bandwidth B, the self-queuing time for the client is RowSorterSize / B. The latency allowance must be more than the self-queuing time to avoid having the arbiter in Latency Mode all of the time. If the row sorter is known to contain requests from other clients with the same latency allowance, the self-queuing time for a given client can be reduced. However, if the row sorter may contain requests from another client with shorter latency allowance (e.g. CPU), the self-queuing time may be extended.
- Request arrival rate. Some clients dribble requests into the row sorter. These requests may have page-locality and may be able to be processed in the same activate, if the row sorter holds onto the initial request(s) long enough for the subsequent requests to arrive in the row sorter. It is desirable to choose a latency allowance that will allow the aggregation of requests of this type.

Consider rules for choosing the latency allowance for different client classes.

For convenience in later sections, the following components of read latency (the numbers in parentheses assume 800 MHz emcclk) are rolled up:

```
static latency from client to snap arbiter        = 11 emcclks   (14ns)
static latency from snap arb to end of row sorter = 4-54 emcclks (5-68ns)[1]
static latency from row sorter to DRAM pads       = 14 emcclks   (18ns)
DRAM latency (assuming 800 MHz DRAM)              = 24 emcclks   (30ns)
static latency from DRAM pads back to client      = 16 emcclks   (20ns)
                                          -------------------------
                                                  = 69-101 emcclks (86-150ns)
A useful rolled up number is the total static latency (including DRAM latency), but excluding the
portion covered by the latency allowance:
static latency minus snap arbiter to row sorter   = 65 emcclks   (81ns)
```

18.8.2.1 Latency Allowance for Write Clients

Because of the WCAM, the latency allowance for write clients is greatly simplified, which required quick scheduling of writes so that write acknowledges could be returned. Once a request has passed the WCAM, even for a low-latency client like CPU, the contents of the write are globally visible. The actual scheduling of the write can be done leisurely. The recommended latency allowance for all write clients is 128 ticks at maximum emcclk. The value should scale up as emcclk frequency scales down, so other clients with frequency-dependent LA values don't have higher LAs than writes. The equation for writes for Tegra X1 is:

latency allowance (emcclk) = min(255, 128*(800 MHz/emcclk))

The following sections discuss latency allowance for reads among the different client classes.

1. Assuming: 1) CPU client, 2) GPU client with TLB miss/PDE hit/PTE hit.

18.8.2.2 Latency Allowance for CPU Read Clients

For CPU read clients, the latency tolerance is dictated by the number of outstanding requests the CCPLEX supports. For Tegra X1, the fast Cortex-A57 multi-core supports a maximum of 11 outstanding reads (whether directly from the CPU or from the L2 prefetcher). The latency allowance is dictated by a tradeoff between a desire to lower latency (thereby increasing bandwidth, according to Little's Law) and a desire to limit the disruptiveness of CPU requests and to allow the memory controller to process streaming CPU reads with more than one access per activate.

18.8.2.3 Latency Allowance for Bandwidth-Soak Read Clients

For a bandwidth-soak read client, latency to the client must be as low as possible (to minimize expensive latency buffering), so the latency allowance must be as low as possible, while still covering the self-queuing time. So start with the self-queuing time and work upward. To avoid continuous latency panic mode, a latency allowance that is larger than the self-queuing time is needed, using conservative estimates of the row sorter drain bandwidth. A target latency allowance is computed at maximum emcclk rate as follows:

```
target latency allowance = row sorter size / conservative drain bandwidth +
                           static latency from snap arbiter to row sorter;
```

Tegra X1 has a 64-entry row sorter per channel. If it is conservatively assumed that a drain bandwidth of 50% of peak DRAM utilization[1] these 2*64 entries will take 128 entries * 4 emc clocks per request / 50% DRAM utilization = 512 emcclks to drain. To this the static latency is added from the ring 2 snap arbiter to the row sorter including the SMMU (assuming TLB miss / PDE hit / PTE hit and that emcclk = 2*mcclk. This adds an additional 54 emcclks. The total is 512 + 54 = 566 emcclks. The number of ticks (and nsec) this corresponds to depends on the mcclk frequency. At 800 MHz emcclk, this corresponds to 708 nsec or 24 ticks.

From the latency allowance, the total latency of the system can be determined, thus the latency tolerance the client and system must support:

```
total latency = static latency minus snap arb to row sorter  ( 65 emcclks) +
                target latency allowance                     (566 emcclks) +
                expiration time                              ( 88 emcclks)
              = 719 emcclks (899ns)
```

The MCCIF response buffer must provide storage for all requests outstanding for the total latency at the required bandwidth. Using Little's Law:

```
mccif size = total latency * required bandwidth
```

where required bandwidth is the maximum sustained bandwidth for the client. For a bandwidth soak client that can saturate memory bandwidth doing reads alone, this would be peak bandwidth derated by the expected DRAM efficiency. Using Tegra X1 numbers gives the following:

```
mccif size = 719 emcclks * 2*16B/emcclk * 0.85 =
          ~= 19.6 KB (306 64B entries[2])
```

Once the mccif size has been determined, the actual latency allowance can be computed by inverting the equations above:

```
latency allowance = (mccif size) / client bandwidth
                    - static latency minus snap arb to row sorter (65 emcclks)
                    - expiration time                             (88 emcclks)
                  = (mccif size) / client bandwidth – 153 emcclks
```

For a bandwidth-soak client:

1. Drain bandwidth may be reduced because of poor locality in the request stream or because the CPU requests with low latency allowance are consuming bandwidth with minimal row sorter occupancy.
2. The number of entries in the MCCIF response buffer is much larger than the row sorter itself, which seems counterintuitive and wasteful—and it is. This buffer does have to cover latency beyond the row sorter latency (for SMMU misses and static latencies in the request/response path outside of row sorter). The main contributor, though, is that latency allowance is calculated based on a conservative estimate of the drain time (based on 50% DRAM utilization), whereas the FIFO must be sized using that latency allowance with best-case bandwidth. Requests can legitimately sit in the row sorter until the latency allowance + expiration time. This wasted MCCIF storage can be avoided if the arbiter had a graduated priority scheme based on increasing age, rather than the current bimodal priority scheme.

```
client bandwidth = emcclk * 16B * 0.85 (efficiency factor)
```

so

```
latency allowance = (mccif size) / (emcclk * 2 * 16B * 0.85) – 153 emcclks
```

For GPU, taking the calculated mccif size of 306 64B entries, this better match up with what is initially calculated at maximum emcclk and it does:

```
latency allowance = 306 * 64B / (800 MHz * 2 * 16B * 0.85) – 153 / 800 MHz
                  = 900 – 191 nsec = 709ns = 24 ticks
```

On Tegra X1, GPU has connects to the MSS with two clients, each with a half sized MCCIF (conservatively sized to 160 entries). Using this MCCIF buffer size, converting to 30 nsec ticks, with the 255 maximum LA clamp taken into account results in the following:

```
latency allowance (ticks) = min(255, 24*(800 MHz/emcclk))
```

Latency allowance is proportional to emcclk period, so at lower emcclk frequencies, latency allowance will increase. This is appropriate looking at the MCCIF capacity alone. However, clients also have an internal latency tolerance based on the size of their internal buffers, number of active warps, etc. If the client clock slows down in proportion to emcclk, then the client's internal latency tolerance will increase as the MCCIF buffer's latency tolerance does. However, if the client is running at high internal clock rate, but generating only limited memory traffic, software could dial the emcclk frequency down, causing the client to become latency-limited. Software addresses this when a bandwidth soak client is enabled by only allowing emcclk to scale down as much as the client clock.

Since client latency requirements are implicitly captured in the emcclk frequency, latency allowance for bandwidth-soak clients can be purely a function of emcclk frequency, with no dependence on the client clock.

At very low emcclk frequencies when latency allowance is clamped at 255 ticks the row sorter drain time approaches and then exceed the client's latency allowance. The client's requests are processed in latency mode, which hurts DRAM efficiency. Activity monitors automatically adjusts emcclk frequency upward based on the lower DRAM utilization, so this situation should be self-correcting.

18.8.2.4 Latency Allowance for Variable-Bandwidth non-ISO Read Clients

Clients, such as VIC, PCIe, SATA, and XUSB have the following characteristics:

- They have a maximum bandwidth requirement that is specified at some emcclk frequency.
- Their bandwidth is allowed to scale back (proportionately) with emcclk below the target frequency.

The VIC read client is one example. If VIC has matched read and write bandwidth, VIC's read bandwidth requirement is half of 85% of peak DRAM bandwidth at maximum emcclk. As emcclk frequency goes down, VIC's bandwidth requirements go down proportionally. The SATA read client is another example. SATA reads have a bandwidth requirement of 300 MB/sec at emcclk = 400 MHz. Bandwidth is allowed to scale down at lower emclk frequencies.

The analysis for these is similar to bandwidth-soak clients, only it is provisioned for the design bandwidth at the target emcclk. As above, MCCIF size is calculated using the following, evaluated at the design bandwidth and target emcclk:

```
mccif size = total latency * required bandwidth
```

which evaluates to:

```
mccif size = 719 emcclks * required bandwidth
```

Latency allowance at the design point can either be calculated based on MCCIF size:

```
latency allowance = (mccif size) / client bandwidth[1]
                    - static latency minus snap arb to row sorter (65 emcclks)
                    - expiration time                             (88 emcclks)
```

or using the target latency allowance expression:

1. Client bandwidth should be expressed in B per emcclk at the target emcclk rate to provide proper emcclk-dependent scaling.

```
target latency allowance =
                    row sorter size / conservative drain bandwidth +
static latency from snap arbiter to row sorter;
```

If the MCCIF size was determined using the expression above, the resulting latency allowances will match. In some cases the MCCIF will be larger than this. Then there is a range of latency allowance choices that are feasible. Choosing the high end of the range (using the MCCIF size expression) gives the arbiter more freedom to schedule other clients. Choosing the low end of the range (using the target latency allowance expression) provides bandwidth headroom above the requirement. The choice can be made on a per-client basis.

Both of these expressions scale latency allowance up as emcclk frequency goes down, as desired. However, at some point as emcclk frequency goes down, LA will reach the maximum LA value of 255 ticks and will be clamped for emcclk frequencies below that threshold. This can create problems because as emcclk goes below the threshold, the row sorter drain time will approach and then exceed the client's latency allowance, meaning that the client's requests will be processed in latency mode If the client's bandwidth is a small fraction of overall bandwidth, this likely isn't a serious problem. It the client's bandwidth is a large fraction of DRAM bandwidth, the client driver may need to require the emcclk frequency to be higher than this cutoff point. Alternatively, the activity monitors may automatically adjust emcclk frequency upward based on the lower DRAM utilization. In either case, some power efficiency will likely be sacrificed.

For emcclk frequencies above the frequency at the bandwidth target, there are two choices: 1) allow LA to scale, which will support higher bandwidth at higher emcclk frequencies, 2) hold LA constant above the target frequency, which will give the arbiter additional headroom to schedule other clients and improve DRAM efficiency at these high clock frequencies. Client requirements should dictate the choice.

These clients are like bandwidth soak clients, only they operate with lower bandwidth and their bandwidth only needs to scale within a portion of the emcclk frequency range. The calculation for two specific clients is shown below.

VIC. VIC must be able to saturate DRAM bandwidth at maximum emcclk, however, VIC reads normally will be balanced with VIC writes, so the VIC read client only needs to around half saturate DRAM bandwidth (assuming 60% reads). MCCIF size is 256 64B entries. So, latency allowance is given by:

```
(mccif size) / (client bandwidth) – 153 emcclks
 = 256*64 / (2*0.6*0.85*16B/emcclk) – 153 emcclks
 = 851 emcclks (28 ticks at emcclk = 800 MHz, linearly scaling down)
```

Expressed in ticks, with the 255 maximum LA clamp taken into account:

```
latency allowance (emcclk) = min(255,  28*(800 MHz/emcclk))
```

XUSB. XUSB has a bandwidth requirement of 880 MB/sec at 300 MHz emcclk. Bandwidth is allowed to taper off at lower emcclk frequencies. MCCIF size is 32 64B entries. So at 300 MHz emcclk, latency allowance is:

```
(mccif size) / (client bandwidth) – 153 emcclks
 = 32*64 / 0.880 GB/sec – 153 * 300 MHz
 = 2327 – 510 nsec = 1817 nsec = 61 ticks
```

This same latency allowance is used at higher emcclk rates and can scale latency allowance up at lower emcclk rates. The proposed expression for latency allowance for XUSB (measured in 30nsec ticks) is:

```
latency allowance (emcclk) = min(255, 61*max(1.0, 300 MHz/emcclk))
```

This expression exceeds the GPU latency allowance at all emcclks, so dropping below the drain time is not a concern.

18.8.2.5 Latency allowance for fixed-bandwidth non-ISO read clients

These clients differ from bandwidth-soak clients in that they have a fixed (use case dependent) bandwidth requirement that does not scale with emcclk, although there is a minimum emcclk required to support a given use case. By not being required to support full memory bandwidth, their MCCIF buffers can be smaller than for bandwidth-soak clients.

As above, the MCCIF buffer is sized using Little's Law:

```
mccif size = total latency * required client bandwidth
```

The maximum of this expression is desired over all client use cases:

```
mccif size = max over all client use cases (
total latency at minimum emcclk that supports use case *
required client bandwidth at that use case)
```

The same value is used for total latency computed above for bandwidth-soak clients (719 emcclks).

The maximum value of this expression will occur often at the maximum client bandwidth. However, since the total latency term depends on emcclk, the maximum value for this expression could occur at a lower bandwidth point. What matters is the ratio of the client bandwidth requirement to the minimum emcclk frequency that supports the use case.

Once the MCCIF response buffer size has been determined, latency allowance for a given use case can be calculated as above:

```
latency allowance = (mccif size) / (client bandwidth) - 153 emcclks
```

When the client bandwidth and emcclk values are not known precisely, the maximum client bandwidth and minimum emcclk frequency should be used.

If the MCCIF was sized based on the maximum expression above, the resulting latency allowance should exceed the row sorter drain time and be acceptable. However, if in the current use case,

```
mccif size < total latency (719 emcclks) * required client bandwidth
```

then the latency of the system must be reduced by one of the following, so the right hand-side of the latency*bandwidth product fits within the actual MCCIF buffer:

1. Increasing emcclk
2. Decreasing the row sorter size

Tegra X1 minimizes the number of performance knobs that have to be changed by software when frequency changes occur. For some NISO clients it may be feasible to choose one latency allowance setting that is valid for all client use cases and emcclk ratios.

18.8.2.6 Latency allowance for ISO read (i.e. display) clients

Display clients differ in several significant ways from the non-ISO clients:

1. Non-ISO clients can tolerate occasional bandwidth disruptions or excess-latency events; average throughput is what matters. Display clients must not underflow.
2. The display internal latency tolerance is essentially unlimited. The memfetch unit and display itself independently track progress through the frame. Memfetch greedily requests all the pixels required for a frame, subject only to back-pressure from MC. The datapath portion of display reads pixels from the response buffer in raster order at the required rate. There is no internal buffer connecting memfetch and display proper that limits the display latency tolerance.

Full-feature display clients (display windows A,B,C)

Full-feature display clients (for windows A, B, C) have two response buffers: a) the MCCIF response buffer, which exists for reordering only, and b) buffering available in the display mempool, which allows display to manage latency disruption events.

The MCCIF response buffer is large enough to accommodate the required display bandwidth times expected latency.

The display total latency is given by the equations above (latency allowance + expiration time + static latencies). The latency allowance must be large enough to satisfy three requirements:

1. Exceed the self-queuing time when DRAM is saturated. This is the same requirement as for the bandwidth-soak and NISO clients. Using the equations in Section 18.8.2.3: Latency Allowance for Bandwidth-Soak Read Clients a latency allowance >= 25 ticks will satisfy this requirement for a client that is saturating DRAM by itself at maximum emcclk. At lower emcclk rates, the self-queuing time varies in proportion to the emc clock period.

2. Be longer than the GPU latency allowance. A premise of the architecture is that display requests will have a relatively long latency allowance, giving the memory controller flexibility to fill in the gaps relative to requests from lower latency clients. Since the GPU latency allowance is 20-25 ticks at maximum emcclk, display latency should be higher, say ≥ 40 ticks at maximum emcclk.
3. Be long enough to group memfetch requests to the same DRAM page so that display requests can be efficient. The goal is to process three 64B requests in a single activate, which maximizes efficiency and minimizes power. This requirement is most difficult to meet for planar and semi-planar YUV surfaces, which have a pixel depth of 1B. At 300 MHz pixclk (3.33ns per clock), it takes 192 YUV pixels * 3.33ns/pix = 640ns to accumulate 192B. This argues that the latency allowance should be >= 21 ticks at maximum pixclk. As pixclk goes down, the time required to accumulate 192B increases. Fortunately, as pixclk goes down, the latency allowance can be increased counteracting this effect.

Requirements 1) and 2) dominate requirement 3). Requirement 1) is used to determine the target latency allowance. In extreme-bandwidth modes, this will result in a display latency allowance that matches the GPU latency allowance. However, in normal (power-critical) modes this will provide margin that will enable the display latency allowance to be larger than the GPU latency allowance.

On Tegra X1, the display MCCIF is sized to 73 to 180 64B entries, depending on the window and its worst use case requirement.

Working backwards from the MCCIF buffer size, the LA is computed as follows:

```
latency allowance = (mccif size) / bandwidth
                  - static latency minus snap arb to row sorter
                  - expiration time
```

The required bandwidth is the active bandwidth, which is roughly equal to the product of: pixclk, bytes per pixel, the horizontal and vertical bandwidth scale factor, and multiplied by 1.1 (margin for catching up after bandwidth disruption events).

Note that several of the terms are emcclk-frequency-dependent and that for a given display bandwidth, these terms are largest at low emcclk frequencies, resulting in smaller latency allowances. Latency allowance must exceed the row sorter drain time, which is largest at low emcclk frequencies. So there is a critical emcclk frequency at which drain time equals latency allowance.This provides a lower bound for emcclk as a function of the display mode.[1] Setting latency allowance equal to row sorter drain time, and solving for emcclk:

```
mccif size / (pixclk*Bpp*hvscalefactor*1.1)
- static_latency_excluding_snap_arb_to_row_sorter__emcclks / emcclk
- expiration_time__emcclks / emcclk
=
row_sorter_size / conservative_drain_bandwidth
+ static_latency_snap_arb_to_row_sorter__emcclks / emcclk
```

Expanding conservative_drain_bandwidth per the equations above and evaluating for emcclk:

```
emcclk = (static_latency_excluding_snap_arb_to_row_sorter__emcclks +
          static_latency_snap_arb_to_row_sorter__emcclks +
          expiration_time__emcclks +
          row_sorter_size / (16B * 0.5))
         * pixclk*Bpp*hvscalefactor*1.1 / mccif_size
```

Given this minimum emcclk frequency, it can be substituted back into the latency allowance equation to determine a latency allowance that will support the required bandwidth at this minimum emcclk frequency:

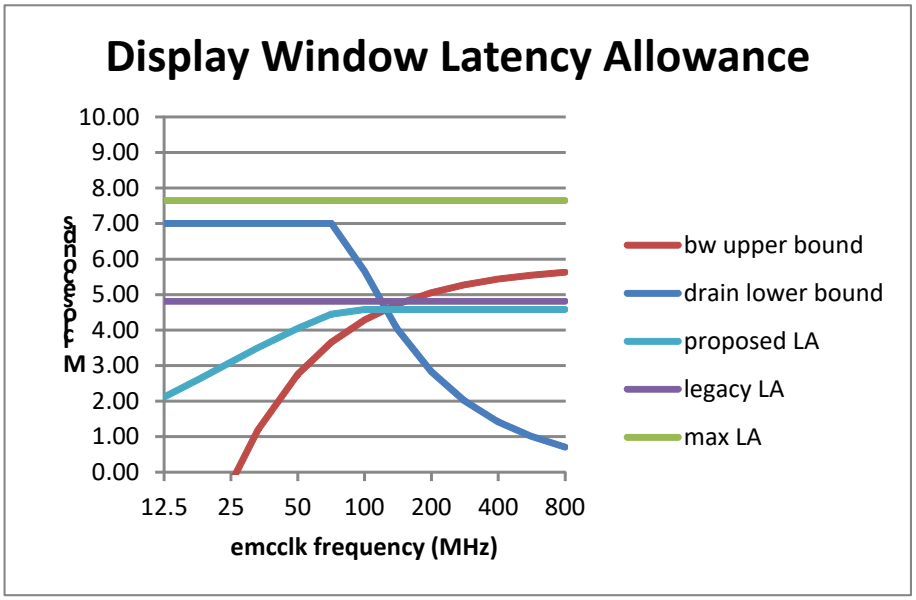
```
latency allowance =
    mccif size / (pixclk*Bpp*hvscalefactor*1.1)
    - static_latency_excluding_snap_arb_to_row_sorter__emcclks / emcclk
    - expiration_time__emcclks / emcclk
```

1. Note that DVFS time also increases with lower emcclk. That also places a lower bound on emcclk, which may be higher or lower than the bound on emcclk computed here. The DVFS-based bound is harder to compute (it cannot be done in a formula). The DVFS bound is ignored here, but this means that a longer latency allowance is determined than necessary.

This latency allowance value can be used at emcclk frequencies higher than the minimum as well. It becomes increasingly conservative as emcclk frequency increases.

The graph below shows the computed latency allowance and the upper and lower bounds for the latency allowance with pixclk = 120 MHz and Bpp = 4. The minimum emcclk frequency occurs at the point the curves cross and the feasible region is to the right of this crossing—below the red curve and above the blue one. For reference, the latency allowance calculated using Tegra X1 equations is also given (it does not always lie within the bounds). Tegra X1 is to use the single latency allowance calculated above, which does not depend on the current emcclk frequency.

Figure 41: Display Window Latency Allowance



Note: *For a given display bandwidth, the latency allowance is smaller at lower emcclk frequencies.*

When lowering emcclk while display is active, the software should program the smaller (more conservative) latency allowance before initiating the frequency change. Similarly, when raising emcclk, software should retain the lower (more conservative) latency allowance used prior to the frequency change until the frequency change is complete.

Under most conditions, bandwidth disruption and DVFS events are not covered by the MCCIF FIFO sizing. Rather, they are accounted for in the 10% excess bandwidth allocated to display clients and in the buffering margin in the display mempool buffer. Certain extreme modes may require more buffering for bandwidth-disruption and DVFS events than can be accommodated in the display mempool. In these cases a portion of the MCCIF buffer may be used for bandwidth-disruption or DVFS buffering. This reduces the effective size of the MCCIF for reordering and streaming. The latency allowance equation above may still be used, but the effective size of the MCCIF should be reduced by the amount used for these other purposes.

Simple display clients (display windows D, T, and cursor)

Simple window display clients (for windows D, T, and cursor) have only a MCCIF response FIFO with no additional response buffering in display. This one buffer must cover: latency allowance + static latency + expiration time + bandwidth disruption time + DVFS time.

Window D can used for applications like the Android status bar and will be enabled in many long-term power scenarios. However, software assures that if WindowD is active, at least one other window will be active. So, for Window D it is assumed that the row sorter drain time and latency allowance are halved relative to the calculation above for Windows A, B, C. Thus, 32 entries (2KB) of MCCIF buffer space is required to allow for streaming at the required bandwidth.

In addition, the window D MCCIF must cover the bandwidth disruption time plus DVFS time. The additional depth required is:

```
max_over_all_emcclk_frequencies(windowD_bandwidth * (bandwidth_disruption_time+DVFS_time))
```

The maximum possible windowD bandwidth at a given emcclk is:

```
min(max_simple_window_bandwidth, max_ISO_bandwidth/2)
```

The max_simple_window_bandwidth term is maximum pixclk (300 MHz) * 4Bpp * 1.1 = 1.32 GB/sec. The max_ISO_bandwidth term can be computed as 16B/emcclk * 40% (the maximum ISO fraction). This is divided by 2, since it is assumed that at least one other window is also active. The bandwidth_disruption_time was calculated as 1342 emcclks.

At high emcclk rates, max_simple_window_bandwidth limits windowD bandwidth, so the additional depth is:

```
1.32 GB/sec * (1342 emcclks + DVFS_time)
```

At low emcclk rates, max_ISO_bandwidth/2 limits windowD bandwidth, so the additional depth is:

```
16B/emcclk * 0.4 /2 * (1342 emcclks + DVFS_time)
```

The first function slopes downward as emcclk frequency increases. The second slopes upward. So the maximum value of the two occurs when max_simple_window_bandwidth equals max_ISO_bandwidth/2 or:

```
1.32 GB/sec = 16B/emcclk * 0.4 / 2
```

Which occurs when:

```
emcclk = 16B * 0.4 / 2 / (1.32GB/sec) = 2.4ns
```

Or, equivalently, when emcclk frequency = 412 MHz. So the worst-case additional buffer depth is:

```
1.32 GB/sec * (1342 * 2.4ns + DVFS_time) =
4.3KB + 1.32GB/sec * DVFS_time
```

The maximum DVFS time is estimated to be <= 2 µs. If it is assumed that DVFS_time is 2 µs, the total required buffer size is:

```
2KB (streaming bandwidth) + 4.3KB + 1.32GB/sec * 2usec =
2KB + 4.3KB + 2.6KB = 8.9KB
```

8.9KB requires 139 64B entries. Note that this is much larger than the MCCIF buffer for a memfetch window. From this, the latency allowance is calculated using the equation above, with an added term for the bandwidth disruption time:

latency allowance =

```
              mccif size(B) / (pixclk(Hz)*Bpp*1.1)
 -    70 / emcclk(Hz)          // static latencies
 -    88 / emcclk(Hz)          // expiration time
              - 1342 / emcclk(Hz)          // bandwidth disruption time
              - dvfs_time
```

Display Window T

Window T may be used in unsecure and secure mode. When used in unsecure mode, it is assumed that windowT has the same bandwidth requirements and software considerations as windowD, so the windowT MCCIF buffer should be at least as large as the windowD buffer (96 entries or 6KB). In secure mode, windowT may be the only window active, so the “other” window’s bandwidth cannot be counted and conveniently divide by 2 in sizing the MCCIF buffer as with windowD. However, the windowT secure mode is of short duration and that extra power usage due to a higher-than-strictly-necessary emcclk will be acceptable. It is best to use the same 96-entry (6KB) MCCIF buffer as for windowD and to run emcclk as high as necessary to achieve a latency allowance that is greater than the drain time. The equation for windowT latency allowance is the same as the one above for windowD.

Display Cursor Window

For cursor, the active bandwidth requirement can be reduced because at most only 256 pixels are visible on a given scanline, so the bandwidth to fetch those pixels can be amortized over the scanline time. It is assumed that the cursor is no wider than 10% of the screen width, thus the cursor bandwidth is 10% of the bandwidth of an active window on the same head and the MCCIF buffer storage required for streaming is 10% that of windows A, B, and C (10% * 4KB = 400B).

There are two ways of handling DVFS for cursor:

1. Cursor handles DVFS by turning off ready_for_latency_event during the active portion of the frame (only a small fraction of frame time, unlike Windows D and T).

   To compute the MCCIF buffer storage required to cover the bandwidth disruption time, the following is evaluate:

   ```
   max_over_all_emcclk_frequencies(bandwidth_disruption_time*windowHC_bandwidth)
   ```

   The bandwidth_disruption_time was calculated as 1342 emcclks.The maximum possible display bandwidth is (16B/emcclk * 40% (the maximum ISO fraction)). Cursor can consume at most 0.1 / (1.0 + 0.1) = 9.1% of this (since the cursor is assumed to cover at most 10% of the window). So the product is 1342 emcclks * 16B/emcclk * 0.4 * 0.1 / (1.0+0.1) = 781B, independent of emcclk.

   Because cursor consumes all of the pixels in scanline in a short burst, whereas the bandwidth is averaging over a whole scanline, an extra margin of one scanline's worth of cursor pixels (maximum of 256 pixels * 4 Bpp = 1KB) is needed in the buffer.

   Summing these together, a total MCCIF buffer size of 400B + 781B + 1KB ~= 2.2KB is calculated. There are some conservative factors in here, such as the 10% cursor size and targeted latency allowance. So, this is rounded down to 2KB or 32 64B entries.

   To calculate latency allowance for the cursor, the MCCIF buffer size is adjusted downward to take into account the bursty unloading of pixels:

   ```
   mccif_size_adjusted(B) = mccif_size (B) – cursor_width * Bpp
   ```

   Taking this MCCIF size as a given, the latency allowance is computed as follows:

   ```
        latency allowance =
    mccif_size_adjusted / (pixclk(Hz)*Bpp*1.1*bw_ratio)
    –   70 / emcclk(Hz)       // static latencies
    –   88 / emcclk(Hz)       // expiration time
                – 1342 / emcclk(Hz)       // bandwidth disruption time
   where bw_ratio = width of cursor / width of screen (which can include hblank time).
   ```

2. Cursor allows DVFS at any time. This is supported by buffering sufficient cursor pixels that DRAM fetches can be stalled for a DVFS time plus the bandwidth_disruption_time plus other factors. The cursor active bandwidth can be provided for t turning off ready_for_latency_event during the active portion of the frame (only a small fraction of frame time, unlike Windows D and T).

18.9 Arbitration Domain

The arbitration domain outside of MC module consists of super and partition clients, as well as any pipe stages necessary to route the partition-client interfaces across the chip.

18.9.1 Module Superclients

Module superclients are relatively simple multiplexes of the requests from all of the MCCIFs in the module.

18.9.2 Partition Clients

Partition clients are similar to Module Superclients in that they are multiplexes of the requests from multiple groups of MSCs.

Each partition client instantiates a snap-arbiter built of a single snap-arbitration ring. Each client gets one slot in that ring. Whenever the partition client has credits for the input FIFO in MC (NV_MC_cif.vx) and it has pending requests, it arbitrates and issues the pending requests.

18.9.3 Snap Arbitration

The Snap Arbiters choose one active client request from across the entire Arbitration Domain.

18.9.3.1 Snap Arbiter Design

Conceptually, each Snap Arbiter works as follows:

1. The Snap Arbiter records a bit vector of all clients with active requests at this point in time.
2. While the bit vector is not clear, the Snap Arbiter selects one of the bits in a fixed priority order.
3. The Snap Arbiter issues the selected client and clears the bit, then continues with step 2.
4. Once all the bits in the previous bit vector are clear, the Snap Arbiter "snaps" another set of requests by going to step 1.

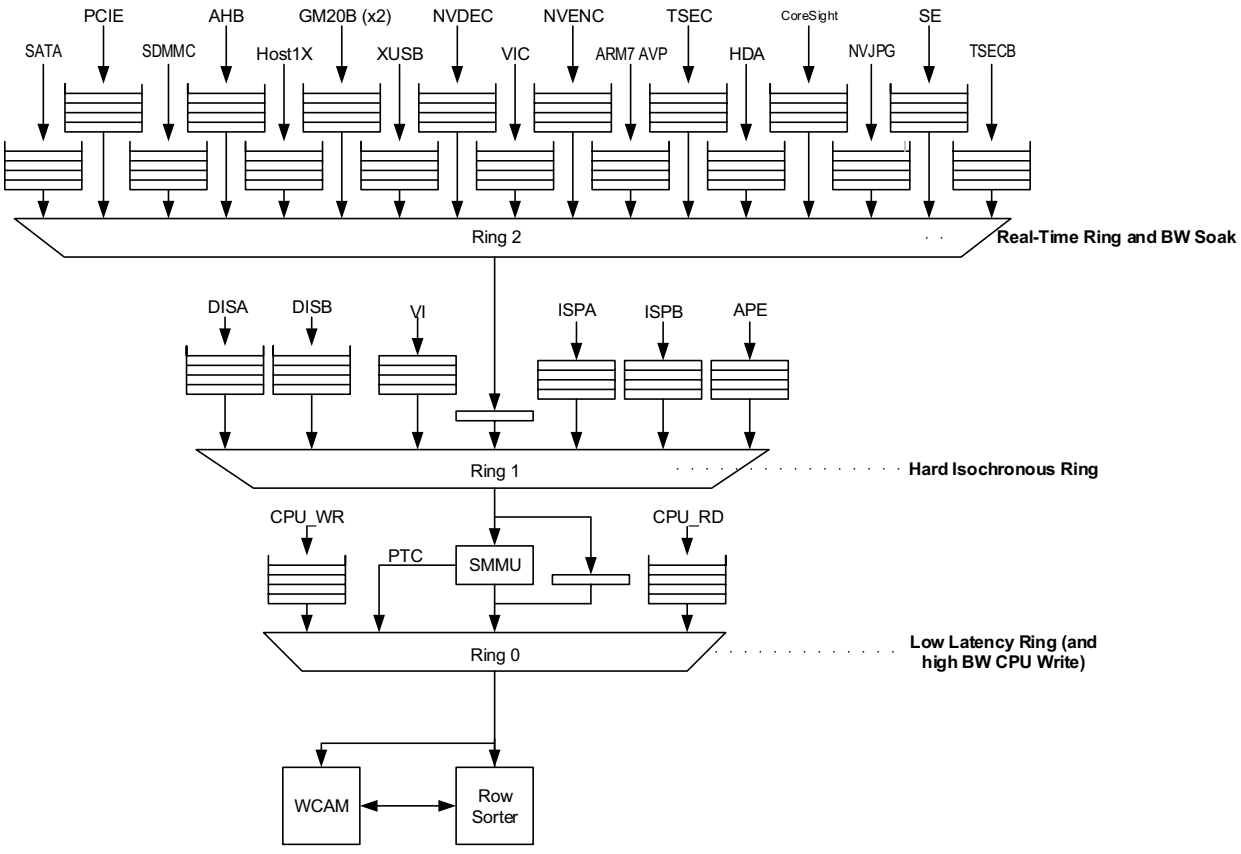
A Snap Arbitration ring makes sure that if N clients are constantly requesting, each gets $^1/_N$ of the available bandwidth (if the weights of all clients are equal).

18.9.3.2 Snap Arbiter Trees

Multiple Snap Arbiters can be chained. For example, if a three-way snap arbitration ring feeds into one of the inputs of a four-way snap arbitration ring, the three clients in the top ring are guaranteed 1/4th of the bandwidth and the three clients in the bottom ring are guaranteed 1/12th of the bandwidth. Note that the snap arbiters do not limit the maximum bandwidth of any client – if a single client is the only requester, it will get 100% of the available bandwidth.

18.9.3.3 The Snap Arbiter Tree

Figure 42: The Tegra Snap Arbiter Tree



The tree is structured in three “rings.”

Ring	Description	Type of Clients	Output Feeds
2	Real-Time Ring	Clients that have real-time requirements but are not isochronous, e.g. MPE. Ideally GPU is a bandwidth soak client. But since that is the only bandwidth soak client left, it is merged into Ring2.	Ring 1
1	Hard-ISOchrononous	Hard ISO clients that have hard latency requirements and fixed bandwidth requirements, e.g. Display, VI, and MSENC	SMMU
0	Low Latency Client	Clients that are latency-sensitive, like PTC and CPU.	Row Sorter

Ideally the Display engine should use carveout memory and not use SMMU translation. This eliminates SMMU translation bottlenecks related to TLB and PTC misses. Display memory in carveout is located in Ring1. Tegra X1 has Display in Ring1 and uses SMMU translation.

18.9.3.4 Bandwidth guarantees for isochronous clients (Priority Tier Snap Arbiter)

In Tegra X1, a snap arbiter ring can be programmed to give multiple grants to the same partition client for a given snap. For example, one would commonly enable 4 display slots in ring 1 and one slot the input from ring 2. Whenever display falls behind in having its requests accepted, it would begin to get more than one grant per snap. When it is most critical, it will get 4 grants for every 1 grant of ring 2 requests. However, there are at least three issues with this scheme.

1. When the efficiency of the memory controller drops, getting 4 grants per snap may still not be enough request bandwidth to keep up with the display demands, eventually resulting in a display underflow.
2. When two display heads are active, the “spool up” (initial fill of data buffers during VBLANK) of one display can take too much bandwidth away from the other display, causing the second display to underflow.
3. The granularity of bandwidth allocation by the Tegra X1 snap arbiter was too coarse, requiring significant “rounding up” for many configs. Although CPU was expected to consume a small fraction of available bandwidth, it could consume up to 1/3. Display in spoolup could starve other ISO clients, such as camera.

These issues are addressed in Tegra X1 by:

- Adding bandwidth guarantee logic to the snap arbiter inputs.
- Providing independent inputs for the display heads into the ring arbiters

The bandwidth guarantee logic uses a simple DDA mechanism in the PTSA, along with dynamic masking of the inputs to the snap arbiter. For each input at any snap arbiter ring, the following control, state, and I/O are provided.

> ***Note:*** *Signed (twos complement) fixed point arithmetic is used for the bandwidth calculations, using F fraction bits and N integer bits (including sign). F=8 and N=6*

Name	Control□state□I/O	Fixed pt format (#int bits . #frac bits)	Description
request	input	-	Client is making a request (true if any slot of the client is requesting)
grant	input	-	Client request was granted last cycle
high_priority	output	-	Client’s request grants are behind the guaranteed rate
rate	control	0.F	Amount to increment accum every mcclk.
accum	state	N.F	Accumulates number of grants “owed” to this client; a negative value indicates that the client has had more grants than its guarantee
maximum	control	N.0	Maximum value of accum allowed
minimum	control	N.0	Minimum value of accum allowed (normally negative)
grant_decrement	global control	1.F	Amount to subtract from accum every mcclk that a grant occurs. This is a global config value that is used for all DDAs.

Each MC cycle, the following computation is done for each client:

```
high_priority = (accum >= 0.0);
accum = accum + rate;
if (grant) accum = accum – grant_decrement;
if (accum > max) accum = max;
if (accum < min) accum = min;
```

If any clients of the ring arbiter have a non-zero value for (high_priority & request), that is, it is high priority and at least one of its slots has a valid request, then the snap arbiter requests inputs with a mask where the mask value is the client's high_priority signal (the client's high_priority is used as a mask for all of the client's slot requests). If no client has a non-zero value for (high_priority & request), then there is no masking of the snap arbiter inputs. Thus any clients that are "behind" will be handled in the next snap arbiter round while clients that are not behind are excluded.

The control parameters *rate*, *min*, and *max* are typically programmed to provide the appropriate bandwidth and operating bounds at maximum mcclk/emcclk with *grant_decrement* =

- 1.203 hex for LP4 with maximum emcclk of 800 MHz
- 1.501 hex for LP3 with maximum emcclk of 933 MHz

At lower mcclk frequencies, rather than increasing each client's rate parameter, the global value grant_decrement is lowered instead.

The PTSA DDA logic is was motivated by the needs of ISO clients, but will be useful for all clients. The DDA logic is to be used as follows for the different client classes:

- **Hard ISO** (display, camera capture, etc.). The PTSA DDAs are used to guarantee bandwidth. *Rate* should be programmed to the steady-state[1] (non-spoolup) bandwidth requirement of the client * 1.1. The extra 10% factor is to allow the client to catch up after DVFS or other service interruption events. Min should be set to -5. The suggested value is rate*32 (mcclks), but the exact setting is not expected to be sensitive. Max should be high enough that with accum hovering near 0 during normal operation, normal MC events, such as PTC misses and refreshes don't result in saturation at max (thus lost bandwidth). The recommended value is 31. The reason for not choosing a larger max value is that it would cause ISO clients to hog the system for an unreasonable time after a bandwidth disruption event.This hogging can harm other ISO clients and will lower the performance of NISO clients by extending the time they are outright denied bandwidth vs. slightly reducing their sustained bandwidth.[2].
- **CPU**. The PTSA DDAs may be used similarly to guarantee a fraction of bandwidth, at low latency, to the CPU. The CPU may request high bandwidth in memcpy, etc. cases, but it requests only a small fraction of memory bandwidth when in latency-critical mode. The CPU read DDA is set to 9% of total MC bandwidth, and the CPU write DDA is set to 1% of total MC bandwidth. Min can be programmed to -4. Max should be programmed fairly low, perhaps around +4, so that the CPU isn't a bandwidth hog after memory system disruptions.
- **Soft-ISO** (mpeg encode, decode, etc.). The PTSA DDAs may similarly be used to request bandwidth. Soft ISO clients are typically in an outer ring without a strict bandwidth allocation, so the DDA doesn't guarantee bandwidth, but it will let these clients compete against non-ISO clients on the ring. For Ring2 soft ISO clients, the DDAs can be programmed with min = max = 1.0 and rate = 0, so they always have high priority. For Ring1 soft-ISO clients (for example, ISP can be configured as Soft-ISO or Hard ISO depending on the use case), they should be programmed the same as Ring2 NISO clients i.e., with min = -2, max = 0, rate = epsilon (i.e., 1). In more nuanced circumstances, Ring 2 soft-ISO clients could be programmed similarly to ISO clients.
- **Non-ISO.** The PTSA DDAs should not be used to guarantee bandwidth. These clients require bandwidth on a best-effort basis. The DDAs, however, can be used to implement a timeout counter, to ensure that these clients get at least one transaction in every N cycles, preventing them from complete starvation if memory efficiency drops and ISO

1. If a client fetches at double-rate for one scan line and does not fetch for the next scan line, *rate* should be set to the double rate, since the DDA cannot average bandwidth over more than a few dozen clocks.
2. There is a tradeoff in the setting of *max*. The larger *max* is, the more bandwidth disruption events will be captured and the better the bandwidth guarantee will be. However, a larger *max* value lengthens the recovery time after a bandwidth-disruption event, when the client continuously requests at high priority, squeezing out bandwidth for NISO clients and potentially hurting other ISO clients. 128 mcclks is a compromise. It should capture PTC misses and normal DRAM stalls due to page-conflicts, etc. It will capture refresh periods so long as they are 320ns or less. Certain high-density DRAMs like 8Gbit DDR3 have longer refresh periods than this so some saturation to *max* and predictable bandwidth loss will occur. This can be handled by slightly increasing the DDA allocation for the client to make up for the loss from *max* saturation, or increasing the *max* value. Another complication is that if *rate* is greater than 0.25, 128**rate* will exceed the maximum *max* value. Either1) scale back *rate* and *grant_decrement* (requiring software to program all the *rate*, *min*, and *max* values on frequency changes), or 2) recognize that *max* will be lower than desired and provision the client with extra bandwidth. The option recommened is 2).

clients would otherwise consume it all. Rate is typically programmed to its minimum value (1 (epsilon). Min is typically programmed to -2 and max to 0.

- **PTC.** This PTSA DDA is used to guarantee high priority for PTC requests. *Rate* should be programmed to 0 and *min* and *max* programmed to 1.0 so that PTC requests always have high priority.
- **Ring2->Ring1 feeder.** This PTSA DDA should be programmed as a non-ISO client and only used to prevent starvation of ring2 clients.
- **Ring1->Ring0 feeder.** This PTSA DDA guarantees bandwidth to ring1 ISO clients. It should be programmed as an ISO client, with *rate* equal to the sum of the ISO client rates in ring1. To allow soft-ISO clients in upper rings to compete against CPU reads, rate can be programmed to additionally include soft-ISO client bandwidth too (this only works if there are no hard-ISO clients on ring 0; otherwise their bandwidth could be compromised). Both translated and un-translated outputs from SMMU share this one DDA to avoid head-of-line blocking, since upstream in SMMU the two paths share a single virtual channel.[1]

Programming each DDA with *rate*=0, *min*=-1, and *max*=-1 will cause the DDAs to have no effect. Also, multiple entries are still supported, it is functional with the DDAs disabled and the snap arbiter operating as it would without the DDAs.

RTAPI Programming

```
struct dda_client {
    string name; //name of client
    float  bw;   // remember client BW requirement in case of frequency change
    hw_reg min;  // DDA min register
    hw_reg max;  // DDA max register
    hw_reg rate; // DDA rate register
}

////////////////////
// Global constants
////////////////////
int PTSA_REG_LENGTH_BITS = 12;
int RING2_DDA_RATE = 12;
float CPU_RD_BW_PERC = 9;
float CPU_WR_BW_PERC = 1;
int DRAM_WIDTH_BITS = 64;
ISO_t CPU_RD_TRAFFIC_TYPE = NISO;
ISO_t CPU_WR_TRAFFIC_TYPE = NISO;
float DDA_BANDWIDTH_MARGIN = 1.10;

// set at init
float LO_GRANT_DEC;
float HI_GRANT_DEC;
float LO_FREQ;
float HI_FREQ;
float BW_AT_LO_FREQ;
unsigned int GRANT_DEC_INT;
unsigned int GRANT_DEC_FRAC;
unsigned int PTSA_GRANT_DEC_FRAC_MASK;
float DDA_DIV;
unsigned int PTSA_RATE_MASK;
////////////////////
```

1. Ring 0 still has separate inputs for SMMU translated and SMMU bypass (untranslated) requests. One DDA covers them both. If both have valid requests they both can be included in one ring 0 snap arbiter round. The DDA accum value is decremented as each request is granted, which occurs on two separate clock cycles.

```
void init() {
    PTSA_GRANT_DEC_FRAC_MASK = (1 << PTSA_REG_LENGTH_BITS) - 1;
    PTSA_RATE_MASK = (1 << PTSA_REG_LENGTH_BITS) - 1;

    if(dram_type == LP4) {
        LO_FREQ = 25;
        HI_FREQ = 2132;
    } else {
        LO_FREQ = 12.5;
        HI_FREQ = 1200;
    }
    BW_AT_LO_FREQ = LO_FREQ * 2.0 * ((float)DRAM_WIDTH_BITS / 8.0);

    HI_GRANT_DEC = ((dram_type == LP3) || (dram_type == DDR3)) ?  (2.0 - pow(2.0, (-1.0 *
(float)PTSA_REG_LENGTH_BITS))) : (1.5 - pow(2.0, (-1.0 * (float)PTSA_REG_LENGTH_BITS)));

    float adj_lo_freq = LO_FREQ / 2.0;
    LO_GRANT_DEC = HI_GRANT_DEC * (adj_lo_freq / (HI_FREQ / 2));

    if(dram_type == LP4) {
        DDA_DIV = 1.0;
    } else {
        DDA_DIV = 2.0;
    }

    foreach client c {
        if(c.name == "ptc" || c.name == "display" || c.name == "displayb") {
            // always high priority
            c.min  = 1;
            c.max  = 1;
            c.rate = 0;
        } else {
            //disable
            c.min  = 0x3f; //-1 in 2s complement
            c.max  = 0x3f; //-1 in 2s complement
            c.rate = 0;
        }
    }
}

//////////////////////////
// Sets the grant decrement
//////////////////////////
void setGrantDec(float dramFreqMhz) {
    float grantDec = 1.0;

    grantDec = LO_GRANT_DEC * (dramFreqMhz / LO_FREQ);

    if(grantDec >= 1.0) {
        grantDec -= 1.0;
        GRANT_DEC_INT = 1;
    }else {
```

```
        GRANT_DEC_INT = 0;
    }

    GRANT_DEC_FRAC  = fraction2dda( grantDec, 1.0, PTSA_GRANT_DEC_FRAC_MASK, 1);
    GRANT_DEC_FRAC &= PTSA_GRANT_DEC_FRAC_MASK;
}

////////////////////////////////////////////////////
// Convert a floating point fraction into a DDA value
////////////////////////////////////////////////////
unsigned int fraction2dda(float fraction, float div, unsigned int mask, bool round_up_or_to_nearest)
{
    //round_up_or_to_nearest determines whether the final calculated DDA rate is to be rounded up or to
nearest.
    //Using this input to the function we can enable rounding to nearest for NISO client DDA rates.
Rounding up for all other cases to be conservative.
    unsigned int dda = 0;
    float f = $fraction / $div;

    for(int i = 0; i < PTSA_REG_LENGTH_BITS; i++)
    {
        f *= 2.0;
        float r = floor(f);
        dda = (dda << 1) | (unsigned int)(r);
        f -= r;
    }
    if (f > 0.0)
    {
        //Do not round up if the calculated dda is at the mask value already, it will overflow
        if (dda != mask){
            if (round_up_or_to_nearest || (f >= 0.5) || (dda == 0)){
                dda++; //to round up dda value
            }
        }
    }

    return dda;
}

float getDramFreqMhz(float emc_freq) {
    if(dram_type == LP4) {
        return (2 * emc_freq);
    } else {
        return emc_freq;
    }
}

////////////////////////////////////
// Returns peak DRAM bandwidth in MBps
////////////////////////////////////
float getMemBandwidthMBps() {
    if(dram_type == LP4) {
```

```
        return (2 * emc_freq * 2.0 * (DRAM_WIDTH_BITS / 8));
    } else {
        return (emc_freq * 2.0 * (DRAM_WIDTH_BITS / 8));
    }
}

//////////////////////////
// Returns CPU rd DDA rate
//////////////////////////
unsigned int getCpuRdDdaRate() {
    float cpu_rd_bw = getMemBandwidthMBps() * CPU_RD_BW_PERC / 100.0;
    if(dram_type == LP4) {
        cpu_rd_bw = cpu_rd_bw / 2.0;
    }
    unsigned int min_perc_rate = fraction2dda(cpu_rd_bw/ BW_AT_LO_FREQ * LO_GRANT_DEC,
                                    DDA_DIV, PTSA_RATE_MASK, (CPU_RD_TRAFFIC_TYPE != NISO));
    return max(1, min_perc_rate);
}

//////////////////////////
// Returns CPU wr DDA rate
//////////////////////////
unsigned int getCpuWrDdaRate() {
    float cpu_wr_bw = getMemBandwidthMBps() * CPU_WR_BW_PERC / 100.0;
    if(dram_type == LP4) {
        cpu_wr_bw = cpu_wr_bw / 2.0;
    }
    unsigned int min_perc_rate = fraction2dda(cpu_wr_bw/ BW_AT_LO_FREQ * LO_GRANT_DEC,
                                            DDA_DIV, PTSA_RATE_MASK, (CPU_WR_TRAFFIC_TYPE != NISO));
    return max(1, min_perc_rate);
}

// Called when emcclk changes
void freq_change (float emcclk_new) {
    setGrantDec(getDramFreqMhz(emcclk_new));
    hw_grant_dec_int(GRANT_DEC_INT);
    hw_grant_dec_frac(GRANT_DEC_INT);

    foreach iso client c {
        program_iso_client_dda(c, c.bw, emcclk_new);
    }

    cpu_read_client.min = 0x3c; //-4 in 2s complement;
    cpu_read_client.max = 4;
    cpu_read_client.rate = getCpuRdDdaRate();

    cpu_write_client.min = 0x3e; //-2 in 2s complement
    cpu_write_client.max = 0;
    cpu_write_client.rate = getCpuWrDdaRate();

    ring2.min = 0x3e; //-2 in 2s complement
    ring2.max = 0;
```

```
    ring2.rate = RING2_DDA_RATE;

    ring1.min = 0x3b; //-5 in 2s complement;
    ring1.max = 31;
    unsigned int sum_iso_dda_rates = 0;
    foreach iso client c {
        sum_iso_dda_rates += c.rate;
    }

    unsigned int ring1_dda_rate = sum_iso_dda_rates + ring2.rate;
    if(dram_type == LP4) {
        ring1_dda_rate = ring1_dda_rate / 2;
    }
    ring1_dda_rate += cpu_read_client.rate;
    ring1_dda_rate += cpu_write_client.rate;
    if (ring1_dda_rate > PTSA_RATE_MASK) {
        ring1_dda_rate = PTSA_RATE_MASK ;
    }
    ring1.rate = ring1_dda_rate;

    foreach niso client c {
        c.min  = 0x3e; //-2 in 2s complement
        c.max  = 0;
        c.rate = 1;
    }

    foreach siso client c {
        if(c.ring == 1) {
            c.min  = 0x3e; //-2 in 2s complement
            c.max  = 0;
            c.rate = 1;
        } else { //ring2
            // always high priority
            c.min  = 1;
            c.max  = 1;
            c.rate = 0;
        }
    }
}

////////////////////////////////////////////////////////////////////////
// Called by client driver or routine above to update iso dda settings
////////////////////////////////////////////////////////////////////////
void program_iso_client_dda(dda_client c, float client_bw_MBps, float emc_freq) {
    c.min  = 0x3b; //-5 in 2s complement
    c.max = 31;

    unsigned int ptsa_grant_dec_mixed = GRANT_DEC_FRAC;
    ptsa_grant_dec_mixed += (GRANT_DEC_INT * (1 << PTSA_REG_LENGTH_BITS));
    c.rate = client_bw_MBps / getDramFreqMhz(emc_freq) * ptsa_grant_dec_mixed / DDA_DIV;
}
```

18.9.3.5 Throttling low-priority CPU reads when ring 1 has high DDA priority

SMMU stalls may allow excess CPU read bandwidth to leak into ring 0, even when ring1 ISO clients were due bandwidth. This can hurt ISO bandwidth goals. To address this the MC_EMEM_ARB_MISC1_0.BLOCK_LP_CPU_RD_IF_SMMU_INP_HP config bit is added. It masks off CPU reads when the CPU DDA indicates low priority, the ring1 feeder DDA indicates high priority, and requests are pending from ring 1. This configuration is recommended for low emcclk cases when high ISO bandwidth fractions are desired. It is not recommended at high emcclk frequencies, because it can restrict CPU's peak read bandwidth.

18.9.3.6 CPU reads may interrupt a snap arbiter round

CPU performance critically depends on latency. CPU read requests may be unnecessarily delayed waiting for the current snap arbiter round to complete. Under certain circumstances a high-priority CPU request may interrupt the current snap arbiter round.

CPU read clients should be connected as the first client in whatever ring they are attached to. This first client will be granted first in a snap arbiter round it participates in. A configuration register determines whether high priority CPU reads can interrupt the current snap arbiter round. The cpu_hi_pri_cnt register field (with allowable settings of 0, 1, 2) specifies how many CPU requests can interrupt the current snap arbiter round (a value of 0 means the feature is disabled). If the feature is enabled, the CPU client has a high priority request and the current snap arbiter round does not contain a CPU request, the CPU request is immediately granted. If the register value is 2, a second high-priority CPU request will be taken next. After the high priority CPU requests have been granted, the round completes normally.

Note: *If another high-priority CPU request is present, it will be snapped in the next round and granted first.*

18.9.4 Fractional Update for Tick Counter

For MC clock periods below 200ns, it is difficult to get a TICK that is close to 30ns (the target length of time for a TICK). The TICK counter is controlled by two fields, CYCLES_PER_UPDATE (the number of MC clks between increment of TICK), and EXTRA_TICKS_PER_UPDATE (the increment amount of TICK, over and above 1). When the MC clk period is less than 30ns, EXTRA_TICKS_PER_UPDATE is usually 0. When the MC clk period is greater than 30ns, CYCLES_PER_UPDATE is 1 and EXTRA_TICKS_PER_UPDATE is non-zero. Note that you can certainly use CYCLES_PER_UPDATE != 1 and EXTRA_TICKS_PER_UPDATE != 0 to make the average TICK length closer to 30ns, but this introduces a considerable amount of "jitter" in the TICK counter for MC clk periods greater than 30ns.

The definition of CYCLES_PER_UPDATE and EXTRA_TICKS_PER_UPDATE includes a 4-bit fractional part each. The following table gives examples of the best programming parameters with and without fractional fields for selected low frequencies.

MC frequency (MHz)	period (ns)	EXTRA_TICKS_PER_UPDATE	CYCLES_PER_UPDATE	avg TICK (ns)	EXTRA_TICKS_PER_UPDATE	CYCLES_PER_UPDATE	avg TICK (ns)
13	76.9	2	1	25.6	1.5625	1.0000	30.0
14	71.4	1	1	35.7	1.3750	1.0000	30.1
20	50.0	1	1	25.0	0.6875	1.0000	29.6
25	40.0	0	1	40.0	0.3125	1.0000	30.5
47	21.3	0	1	21.3	0.0000	1.4375	30.6
50	20.0	0	1	20.0	0.0000	1.5000	30.0
83	12.0	0	2	24.1	0.0000	2.5000	30.1
116	8.6	0	3	25.9	0.0000	3.5000	30.2
150	6.7	0	5	33.3	0.0000	4.5000	30.0

18.9.5 Input throttling

Tegra X1 provides a variety of controls to selectively throttle the input request stream. The general motivation for this is to keep a DRAM-limited system from being clogged with requests so it is unresponsive to subsequent high-priority requests. More specifically:

1. **To limit the number of requests in the row sorter at low clock frequencies** so the row sorter drain time doesn't exceed latency allowance. DRAM bandwidth scales down with decreasing clock frequency. At low clock frequency, if the row sorter maintains its full size and the system is DRAM limited, the row sorter drain time can exceed the latency allowance causing the arbiter to run in latency mode all the time, reducing efficiency. It is normally preferable to increase latency allowance and use the full row sorter, but this is not always possible.
2. **To prevent the row sorter from back-pressuring ring 0 when the DRAMs are saturated**. This improves CPU latency and thereby improves CPU (and system) performance (the CPU latency vs. DRAM load curve shows sharply higher CPU latency when the row sorter back pressures).
3. **To avoid filling the HUM when the DRAMs are saturated**. The purpose of the HUM is to prevent head-of-line blocking on an unusually bank queue, not to provide a repository to pile up requests when the row sorter is full. Gratuitous HUM usage perturbs the order requests enter the row sorter and wastes power.
4. **To avoid filling the SMMU when the DRAMs are saturated.** Since the SMMU does not provide a separate virtual channel for ISO requests, this is important to minimize the number of NISO requests ahead of ISO requests that need to get through. It also is important to limit the time requests sit in the system after receiving a deadline and before the row sorter can process them. In low clock frequency cases, delays in SMMU can cause requests to hit the deadlock_avoidance_threshold before they can be granted by the row sorter.
5. **To prevent ring 2 requests from fully occupying the row sorter and SMMU**, leaving row sorter slots available for ISO requests. The concern is that GPU (or other ring 2) traffic, with arbitrarily bad locality can fill the row sorter during temporary lulls in ISO traffic (for example during horizontal blank periods). When display requests resume, the system may not provide enough bandwidth to allow these requests into the row sorter, starving display. Simulations show exactly this behavior in which ISO traffic into ring 1 is delayed due to the row sorter getting filled (during hblank) with poorly behaved 3D traffic.
6. **To prevent non-ISO requests from fully occupying the row sorter and SMMU**, leaving row sorter slots available for ISO and soft-ISO requests (this is similar to issue 5). Simulations show that soft-ISO requests from VDE and MSENC can be swamped by 3D requests.

Throttling may be performed at three locations:

- Output of RING1 (controlled by RING1_THROTTLE)
- Output of RING2 (controlled by the misnamed RING3_THROTTLE)
- Input of RING2 for particular clients (controlled by NISO_THROTTLE)

Throttling is triggered when the number of requests past a designated point in the system (and not granted by the row sorter) exceeds a programmable threshold. A separate threshold is provided for each throttling location.

18.9.5.1 Outstanding transaction counters

The hardware maintains two counters of outstanding transactions that are used in making throttling decisions:

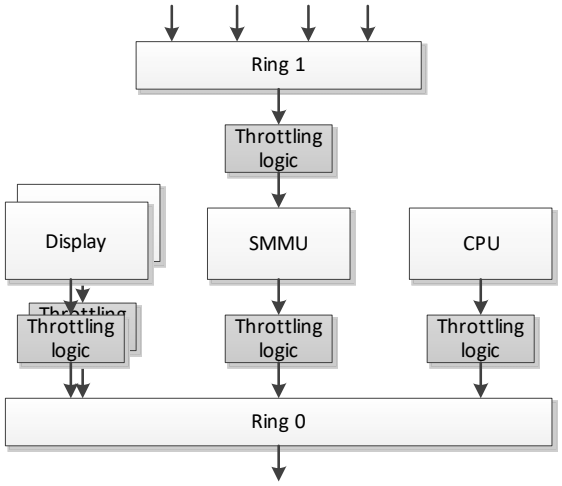
- ARB_OUTSTANDING_COUNT counts the number of requests that have been accepted by ring 0 and are still outstanding in MC. This is exposed in register field: MC_EMEM_ARB_OUTSTANDING_REQ_0.ARB_OUTSTANDING_COUNT.
- ARB_OUTSTANDING_COUNT_RING3 counts the number of requests that have been accepted by any ring and are still outstanding in MC. Note that this includes transactions all of the transactions that would be counted above, but additionally includes transactions in the snap arbiter tree, SMMU, and WCAM. The SMMU has a 64-entry FIFO so it can potentially have up to 64 transactions outstanding. This is exposed in register field: MC_EMEM_ARB_OUTSTANDING_REQ_RING3_0.ARB_OUTSTANDING_COUNT_RING3.
- ARB_OUTSTANDING_COUNT_NISO is an alias for ARB_OUTSTANDING_COUNT_RING3. It has its own output read register field: MC_EMEM_ARB_OUTSTANDING_REQ_RING3_0.ARB_OUTSTANDING_COUNT_RING3.

18.9.5.2 Throttling at the output of Ring 1 (RING1_THROTTLE)

The MC_EMEM_ARB_OUTSTANDING_REQ_0.ARB_MAX_OUTSTANDING threshold is compared with either ARB_OUTSTANDING_COUNT or ARB_OUTSTANDING_COUNT_RING3, depending on config bit: MC_EMEM_ARB_RING0_THROTTLE_MASK_0.ARB_OUTSTANDING_COUNT_ABOVE_SMMU. If disabled, ARB_OUTSTANDING_COUNT is used; if enabled, ARB_OUTSTANDING_COUNT_RING3[1] is used. This comparison determines the throttling level for the following interfaces, as shown in the figure below:

- display input to ring 0
- CPU input to ring 0
- SMMU input to ring 0
- output of ring 1

Figure 43: Conceptual view of ring0 throttle logic



The MC_EMEM_ARB_RING1_THROTTLE_MASK register selects which of the above interfaces is throttled. The degree of throttling is controlled by the MC_EMEM_ARB_RING1_THROTTLE_0 register. The PROD setting for this register are highlighted in bold. Display does not connect to ring 0 in production systems. Throttling was found to be more beneficial at the output of ring 1 than the SMMU input to ring 0 because the latter led to large buildups of requests in the SMMU in DRAM-limited situations, leading to expiry of requests before the row sorter and deadline_avoidance flushes.

This control determines the effective row sorter size and can be used to address issues 1) and 2). It should be set to an emcclk-dependent, empirically-determined value that should permit the full row sorter to be used at moderate to high emcclk frequencies and to reduce the size of the row sorter at lower frequencies. The current recommended formula is:

```
ARB_MAX_OUTSTANDING = CEILING(MIN(Row_sorter_size__B,
                                  Latency_tolerance_ramp,
                                  Deadlock_avoidance_ramp) / 64);
```

where:

```
  Latency_tolerance_ramp = 2*DRAM_width__B*(emcclk__MHz+50);


  Deadlock_avoidance_ramp =
      (Max_latency_usec*emcclk_MHz-Static_latency_snap_arb_to_row_sorter__emcclks)*
2*DRAM_width__B*Conservative_DRAM_efficiency;
```

1. This is the PROD setting, since it includes transactions in the SMMU, which otherwise can grow to an unwieldy number and even expire before entering the row sorter.

and:

```
  Row_sorter_size__B = 4096;          // row sorter size in bytes
  Max_latency_usec = 10;              // max latency
                                      // deadlock avoidance)
  Static_latency_snap_arb_to_row_sorter__emcclks = 54; // current best info
  DRAM_width__B = 8;                  // 4 for a 32-bit memory system
  Conservative_DRAM_efficiency = 0.5; //
                                      // this efficient almost all of the time)
```

The formula has 3 regimes:

1. emcclk > ~200 MHz: Use the full row sorter.
2. ~19 MHz < emcclk <= 200 MHz: Scale MAX_OUTSTANDING down with frequency following a linear ramp empirically to avoid slowdowns to exceeding client latency tolerance.
3. emcclk <= 19 MHz: Scale the row sorter down to keep drain time below 10 µs.

The row sorter needs to be full size in high-emcclk cases (it is paid to sort many client request streams). At some point drain latency gets excessive for certain clients and they start losing throughput due to latency, even though DRAM utilization potentially could be high. At the bottom end, the drain time to keep from triggering deadlock_avoidance timeouts must be clamped.

Note: *The drain time will exceed the maximum latency allowance for lower clock frequencies, meaning the chip will run in latency mode most of the time. This has been found preferable to running with a very small row sorter that is not able to capture the coherence in a simple input stream.*

There is room to optimize this equation within the upper bound from the deadlock avoidance timeout and a lower bound of scaling MAX_OUTSTANDING down linearly with frequency (holding latency constant).

When hysteresis holdoff is allowed, ARB_MAX_OUTSTANDING is overridden and the row sorter is allowed to completely fill (subject to latency allowance expiry). This increases the power savings in low-frequency WinIdle cases when only ISO clients are active. The hysteresis holdoff feature prevents the row sorter from entirely filling in this low-power mode of operation.

18.9.5.3 Output of Ring 2 (RING3_THROTTLE / ARB_MAX_OUTSTANDING_RING3)

The MC_EMEM_ARB_OUTSTANDING_REQ_RING3_0.ARB_MAX_OUTSTANDING_RING3 threshold is compared with MC_EMEM_ARB_OUTSTANDING_REQ_RING3_0.ARB_OUTSTANDING_COUNT_RING3, which counts the number of requests that have been accepted by any snap arbiter ring and are still outstanding in MC. This comparison determines the throttling level for the output of ring 2. The degree of throttling is controlled by the MC_EMEM_ARB_RING3_THROTTLE_0 register.

This control is used to limit the number of requests that can accumulate in the SMMU and to ensure there is space for ISO requests (addresses issues 4 and 5 above). It should be set to an empirically-determined value that is slightly larger than ARB_MAX_OUTSTANDING (suggestion is ARB_MAX_OUTSTANDING + 16).

Note: *PROD settings for RING1_THROTTLE include the SMMU as well, so RING3_THROTTLE is not actually used.*

18.9.5.4 Input of Ring 2 (NISO_THROTTLE / ARB_MAX_OUTSTANDING_NISO)

The MC_EMEM_ARB_OUTSTANDING_REQ_NISO_0.ARB_MAX_OUTSTANDING_NISO threshold is compared with MC_EMEM_ARB_OUTSTANDING_REQ_NISO_0.ARB_OUTSTANDING_COUNT_NISO (the same counter as OUTSTANDING_COUNT_RING3 above). This comparison determines the throttling level for “NISO” partition clients. These are non-ISO ring2 partition clients identified by the MC_EMEM_ARB_NISO_THROTTLE_MASK_0 register. Throttling gates off the

partition client request signals of these clients so the snap arbiter doesn't see the clients' requests. Requests from non-throttled PCs are allowed to proceed. The degree of throttling is controlled by the MC_EMEM_ARB_NISO_THROTTLE_0 register.

This control can be used to address issue 6 above. This is used as RING3_THROTTLE was used on prior chips. When soft-ISO clients are active, non-ISO clients should be identified using the NISO_THROTTLE_MASK register and ARB_MAX_OUTSTANDING_NISO should be set to an empirically-determined value that is higher than ARB_MAX_OUTSTANDING and lower than ARB_MAX_OUTSTANDING_RING3.

18.9.5.5 Limitations of Throttling

Throttling can prevent the row sorter and SMMU from being congested with requests and frequent HUM FIFO when the DRAMs are backed up, but it can limit the number of requests in the row sorter and SMMU when the memory system is operating at high velocity. This can reduce the row-sorter's ability to select efficient requests and can limit the number of requests in flight in the SMMU. The throttle settings should be tuned during bring up. No setting will be perfect for all regimes.

18.9.5.6 CPU Hammer

One limitation found in the mechanisms above is that stalls from PTC misses in the SMMU can allow CPU reads to slip into ring0 beyond CPU's bandwidth allocation, even when ring1 is receiving less bandwidth than programmed. This extra CPU bandwidth reduces the amount of ISO bandwidth the system can provide. To address this, the following additional knob, called "CPU Hammer", is provided:

MC_EMEM_ARB_MISC1_0. BLOCK_LP_CPU_RD_IF_SMMU_INP_HP

When set, it suppresses CPU requests if:

- CPU is ahead of its DDA allocation
- Ring1 is behind its DDA allocation
- Requests are pending at the input of SMMU

When enabled, CPU hammer does block the extra CPU requests in the scenario above. But it can overreach and limit CPU when there isn't enough ring1 bandwidth to saturate the system and the extra CPU bandwidth would have been acceptable (and beneficial).

Unit level testing shows that CPU hammer can raise the ISO bandwidth fraction about 10% (from 46% to 56% at 800 MHz emcclk, with a similar improvement at lower emcclk frequencies). The recommendation to software is to set CPU hammer when ISO bandwidth is needed beyond the low threshold, but to turn CPU hammer if it is not needed for ISO. For this to work, the clocks driver needs to know what the ISO bandwidth demand is and thus can determine whether to set CPU Hammer when selecting a new emcclk frequency.

18.9.6 Display buffer thresholds and "Is Mode Possible" calculations

Full-feature display clients (clients for windows A, B, and C, but not simple windows D, T, and cursor) have four types of buffering available for pixel response data:

1. The portion of the display internal mempool required to buffer line data to avoid refetch (due to the fetch atom being larger than one pixel in height/width or due to scaling).
2. The remainder of the display internal mempool, which can hold pixel data in advance of when it is needed, providing coverage for DVFS or othe r service disruptions.
3. The portion of the MCCIF response buffer that provides storage for requests outstanding in MC (for correctness and reordering).
4. The remainder of the MCCIF response buffer, which can also provide buffering for DVFS or other service disruptions.

The MCCIF response buffer is sized to provide type 3 buffering, but not type 4. The sizing and latency allowance calculations are in Section 18.8.2.3: Latency Allowance for Bandwidth-Soak Read Clients. The display internal mempool (type 2 buffering) provides a cushion for unusually long response latencies and bandwidth disruptions.

For a given display mode to be possible, all four buffer usages must fit within the MCCIF and mempool buffers.To simplify this problem, several conceptual threshold levels are defined within the mempool buffer:

- **threshDisp (“Display threshold”).** Amount display buffer required for the display internal purposes (*e.g.* prefetched blocklinear or rotated display scanlines). This is obtained from the display team.
- **threshLWM (“Low Water Mark Threshold”).** Amount of display buffer to cover display needs, plus bandwidth disruption events.
- **threshDVFS (“DVFS Threshold”).** Amount of display buffer to cover the needs of items above plus span DVFS latency.
- **threshHWM (“High Water Mark Threshold”).** Threshold where buffer occupancy is good enough and there is no benefit to buffering more.

To compute threshLWM, start with threshDisp and add a margin to cover the non-DVFS bandwidth-disruption events.

margin(ns) = max(bandwidth_disruption_time, total_latency)

margin(bytes) = margin(ns) * nominal window bandwidth

The total_latency term can be evaluated using the expression (static latency + latency allowance + expiration time). That leads to an awkward circular dependency for simple windows, whose latency allowance depends MCCIF size, which depends on this margin term. A conservative way to break this dependency is to replace latency allowance with the target latency allowance defined in Section 18.8.2.3: Latency Allowance for Bandwidth-Soak Read Clients which depends on the row sorter drain time and static latencies, and is guaranteed to be larger than the final latency allowance for the mode to be possible.

The next portion of the display buffer is allocated to DVFS, placing threshDVFS the appropriate distance above. To determine the size of the DVFS region:

1. Use the global worst-case DVFS time (~5 µs).
2. Use the worst-case DVFS time for a transition within the range of frequencies the display mode allows.
3. Use the worst-case DVFS time for a transition from the current mcclk frequency to any frequency it could transition to.
4. Use the DVFS time for a transition from the current frequency to a particular frequency being requested.

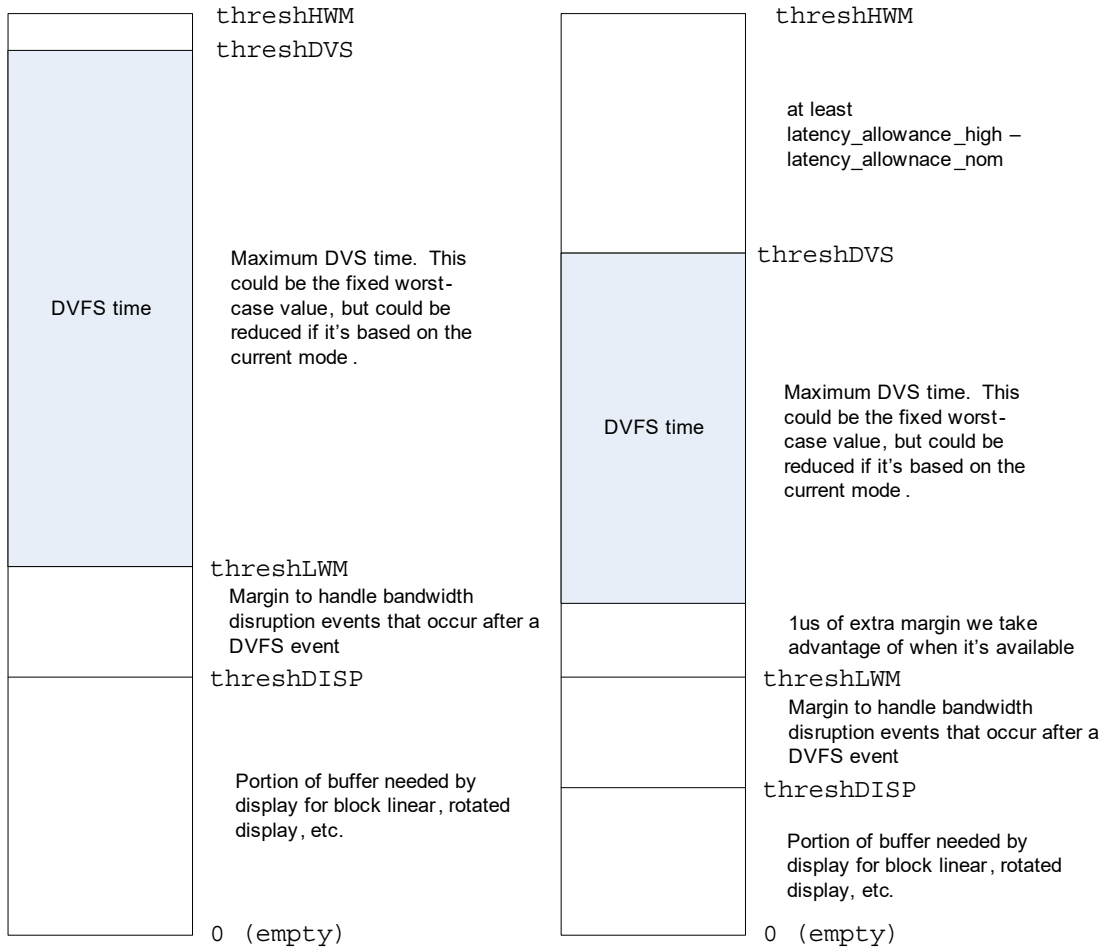
Software can decide which of the above criteria to use. As with margin, to determine the buffer distance in bytes, multiple the DVFS region size in nsec by the nominal window bandwidth.

If either threshLWM or threshDVFS is higher than the top of the buffer, the mode is not possible. For certain extreme modes that don’t require the full MCCIF for reordering/streaming, a portion of the MCCIF buffer can be used to cover bandwidth-disruption events.[1]

If threshDISP is far from the top of the buffer, it is recommended moving it up ~1 µs to provide extra margin after a DVFS event and to make it less likely that the low water mark would be reached. The left figure (a) below shows a high-bandwidth configuration, in which the display requirements, margin, and DVFS require the entire buffer. The right figure (b) shows a more conservative configuration possible when bandwidth requirements are lower. Extra margin is provided above threshLWM and threshDVFS is not at the top of the buffer. In a configuration like this, it would be possible to leave threshDVFS near the top of the buffer, but that would unnecessarily delay the start of DVFS after a vblank, which could be a large fraction of a frame time when a window with a large mempool (capable of handling rotation) is used in a low resolution mode.

1. In such cases, set threshDVFS to the top of the mempool buffer and calculate the amount of additional buffering needed and subtract this from the size of the MCCIF when calculating the latency allowance for the window. If the resulting latency allowance is too short (smaller than the DRAM drain time) the mode is not possible. If the mode is possible, note that the `ready_for_latency_event` signal will not take into account buffered in the MCCIF buffer. Thus, if `ready_for_latency_event` is deasserted, the system is not ready to handle DVFS. However, if `ready_for_latency_event` is asserted, the system may not have sufficient data buffered to robustly handle the bandwidth outage. In these modes, software should separate DVFS events sufficiently to allow buffer recovery from the prior DVFS event.

Figure 44: Display Mempool Thresholds



18.10 Refresh Overhead

The various DRAM types have significantly varying refresh parameters, thus refresh overhead. The following table summarizes the differences:

DRAM type	Density	Temp (°C)	tRP ns	tRCD ns	Refresh interval: tREFI µs	Refresh time: tRFC ns	Refresh overhead (throughput) (tRP+tRFC+tRCD)/ tREFI percent	Refresh overhead (latency): (tRP+tRFC+tRCD) /(2*tREFI) percent	Refresh overhead (latency): $(tRP+tRFC+tRCD)^2$ /(2*tREFI) nsec
LPDDR3-1866	4Gb	0-85	27	24	3.9	130	4.64%	2.32%	4.2
	8Gb	0-85	27	24	3.9	210	6.69%	3.35%	8.7
	4Gb	85-95	27	24	1.95	130	9.28%	4.64%	8.4
	8Gb	85-95	27	24	1.95	210	13.38%	6.69%	17.5
LPDDR4-3200	4Gb	0-85		18	3.9	130	3.79%	1.90%	2.8
	8Gb	0-85	21	18	3.9	180	5.62%	2.81%	6.1
	4Gb	85-95	21	18	1.95	130	8.67%	4.33%	7.3
	8Gb	85-95	21	18	1.95	180	11.23%	5.62%	12.3

18.11 Memory Controller Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

18.11.1 MC Registers

USAGE NOTE: Many MC register fields are shadowed.

Writes to shadowed register fields update the shadow copy (this is default, assumes TIMING_CONTROL_DBG.WRITE_MUX==ASSEMBLY).

Reads to shadowed register fields return the active copy (this is default, assumes TIMING_CONTROL_DBG.READ_MUX==ACTIVE).

This allows a new set of frequency-dependent timing parameters to be written to the shadow registers while memory traffic is ongoing, then when the parameters are completely written, the MC hardware can perform a timing-safe switch.

Such switches can be triggered via two methods: TIMING_CONTROL.TIMING_UPDATE (generally used during initialization), or the automatic CAR/EMC handshake on a clock-frequency or divider change (assuming TIMING_CONTROL_DBG.IGNORE_EMC_TRIGGERS==DISABLED).

Registers that are shadowed are marked by a comment:

> ***Note:*** *This register is shadowed: see usage note in Section 18.11.1: MC Registers.*

USAGE NOTE: Many MC registers should be programmed by the Boot Loader as part of the warm boot (also known as "wake from LP0") and cold boot (also known as "power up") sequences. Suggested actions for the Boot ROM/Boot Loader are noted under "Boot requirements:".

USAGE NOTE: Some MC registers may only be accessed by TrustZone-secured register transactions. See the appropriate ARM Architecture Reference Manual for information on how to handle security. Such registers are marked by a comment:

“Access to this register is restricted to TrustZone-secured requestors.”

USAGE NOTE: The MC register fields to be stored in PMC scratch registers for warm boot must have "[PMC]" in their comments. After adding and removing such PMC flag, the tool warmboot_code_gen must be rerun to generate new boot ROM code.

USAGE NOTE: When writing to a register, any bits that are not intended to be modified should be rewritten with their existing values.

USAGE NOTE: Unspecified bits may not appear in tables and should be written with their Reset values.

18.11.1.1 MC_INTSTATUS_0

Interrupt Status

This register contains trigger bit fields. Bit fields not shown here are reserved.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00xx000000x0xxxxx)

Bit	Reset	Description
17	CLEAR	DECERR_GENERALIZED_CARVEOUT_INT: 0 = CLEAR 1 = SET
16	CLEAR	DECERR_MTS_INT: This interrupt is for access violation on MTS carveout region in the MC. If a client other than CPU read or write is trying to access the region. 0 = CLEAR 1 = SET

Bit	Reset	Description
13	CLEAR	SECERR_SEC_INT: The request violated the SEC Carveout requirements - Only TSEC clients may access the SEC carveout. 0 = CLEAR 1 = SET
12	CLEAR	DECERR_VPR_INT: The request violated the VPR requirements - in case of a read, the request address fell in the VPR range, but the VPR attribute of the request was not set. In case of a write, the request had the VPR attribute set in the request, but the request address did not fall in the VPR range. 0 = CLEAR 1 = SET
11	CLEAR	INVALID_APB_ASID_UPDATE_INT: ASID Update through APB interface resulted in an error. APB interface tried to update secure ASID through Non-secure interface. 0 = CLEAR 1 = SET
10	CLEAR	INVALID_SMMU_PAGE_INT: Address translation error: An access was attempted to an invalid page or a protected page. See ERR_STATUS and ERR_ADR for more information. 0 = CLEAR 1 = SET
9	CLEAR	ARBITRATION_EMEM_INT: Warning that a pending request has reached the deadlock-prevention slack threshold. This indicates that the MC could not meet the programmed performance goals. This interrupt is for use with assisting debug of performance-related issues. 0 = CLEAR 1 = SET
8	CLEAR	SECURITY_VIOLATION_INT: Address translation error: A nonsecure access was attempted to a secured region. See ERR_STATUS and ERR_ADR for more information. 0 = CLEAR 1 = SET
6	CLEAR	DECERR_EMEM_INT: Address translation error: EMEM Address Decode Error. See ERR_STATUS and ERR_ADR for more information. 0 = CLEAR 1 = SET

18.11.1.2 MC_INTMASK_0

Interrupt Masks

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot. Bit fields not shown here are reserved.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xx000000x0xxxxx)

Bit	Reset	Description
17	MASKED	DECERR_GENERALIZED_CARVEOUT_INTMASK: 0 = MASKED 1 = UNMASKED
16	MASKED	DECERR_MTS_INTMASK: 0 = MASKED 1 = UNMASKED
13	MASKED	SECERR_SEC_INTMASK: Prevents SECERR_SEC_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
12	MASKED	DECERR_VPR_INTMASK: Prevents DECERR_VPR_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
11	MASKED	INVALID_APB_ASID_UPDATE_INTMASK: Prevents INVALID_APB_ASID_UPDATE_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
10	MASKED	INVALID_SMMU_PAGE_INTMASK: Prevents INVALID_SMMU_PAGE_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED

Bit	Reset	Description
9	MASKED	ARBITRATION_EMEM_INTMASK: Prevents ARBITRATION_EMEM_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
8	MASKED	SECURITY_VIOLATION_INTMASK: Prevents SECURITY_VIOLATION_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED
6	MASKED	DECERR_EMEM_INTMASK: Prevents DECERR_EMEM_INT from triggering an interrupt to the interrupt controller. 0 = MASKED 1 = UNMASKED

18.11.1.3 MC_ERR_STATUS_0

Error Data Capture: Status

If a translation error is detected (for example, address-out-of-bounds, security fault, protection fault), this register records information about this access:

- which fault the captured error corresponds to
- a read/write indicator and the request ID (global client ID assigned by the memory controller, see the list below)

If the error was an INVALID_SMMU_PAGE, then information about the page's protection status is also captured:

- whether the page was marked non secure in the page-table
- whether the page was marked readable in the page-table
- whether the page was marked writeable in the page-table

Note: *SMMU page protection bits are formed by ANDing the page protection bits from the Page Table Base, Page Directory Entry, and Page Table Entries from all three stages of the page-walk. The ANDed protection bits and the type of the provoking transaction are both available in the status register.*

Subsequent faults (of any type) will not change the status and address registers until the corresponding interrupt is cleared.

The global memory client IDs are defined as follows:

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csr_ptcr	0	(0x00)	ptc				Misses from System Memory Management Unit (SMMU) Page Table Cache (PTC)
csr_display0a	1	(0x01)	dc		display	dis	Display reads, window A
csr_display0ab	2	(0x02)	dcb		displayb	disb	Display reads, window A
csr_display0b	3	(0x03)	dc		display	dis	Display reads, window B
csr_display0bb	4	(0x04)	dcb		displayb	disb	Display reads, window B
csr_display0c	5	(0x05)	dc		display	dis	Display reads, window C
csr_display0cb	6	(0x06)	dcb		displayb	disb	Display reads, window C
csr_afir	14	(0x0e)	afi		afi	pcx	PCIe reads

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csr_avpcarm7r	15	(0x0f)	avpc		avpc	avp	ARM7 BPMP-Lite reads via the avp_cache
csr_displayhc	16	(0x10)	dc		display	dis	Display reads, cursor
csr_displayhcb	17	(0x11)	dcb		display	disb	Display reads, cursor
csr_hdar	21	(0x15)	hda		hda	hdapc	High-definition audio (HDA) reads
csr_host1xdmar	22	(0x16)	hc	hc1	host1x	host	Host channel data reads
csr_host1xr	23	(0x17)	hc		host1x	host	Host indirect reads
csr_nvencsrd	28	(0x1c)	nvenc		nvenc	mse	
csr_ppcsahbdmar	29	(0x1d)	ppcs		ppcs	ahb	AHB DMA engine reads
csr_ppcsahbslvr	30	(0x1e)	ppcs	ppcs1 ppcs2	ppcs	ahb	AHB bus reads
csr_satar	31	(0x1f)	sata		sata	sax	SATA reads
csr_mpcorer	39	(0x27)	cpu		cpu	ftop	Reads from Cortex-A9 4 CPU cores via the L2 cache
csw_nvencswr	43	(0x2b)	nvenc		nvenc	mse	
csw_afiw	49	(0x31)	afi		afi	pcx	PCIe writes
csw_avpcarm7w	50	(0x32)	avpc		avpc	avp	Arm7 BPMP-Lite writes via the avp_cache
csw_hdaw	53	(0x35)	hda		hda	hdapc	High-definition audio (HDA) writes
csw_host1xw	54	(0x36)	hc		host1x	host	Host indirect writes
csw_mpcorew	57	(0x39)	mpcore		mpcore	ftop	Writes from Cortex-A9 4 CPU cores via the L2 cache
csw_ppcsahbdmaw	59	(0x3b)	ppcs		ppcs	ahb	AHB DMA engine writes
csw_ppcsahbslvw	60	(0x3c)	ppcs	ppcs1 ppcs2	ppcs	ahb	AHB bus writes
csw_sataw	61	(0x3d)	sata		sata	sax	SATA writes
csr_ispra	68	(0x44)	isp2		isp2	isp	ISP Read client for Crossbar A
csw_ispwa	70	(0x46)	isp2		isp2	isp	ISP Write client for Crossbar A
csw_ispwb	71	(0x47)	isp2		isp2	isp	ISP Write client Crossbar B
csr_xusb_hostr	74	(0x4a)	xusb_host		xusb_host	usbx	XUSB reads
csw_xusb_hostw	75	(0x4b)	xusb_host		xusb_host	usbx	XUSB_HOST writes
csr_xusb_devr	76	(0x4c)	xusb_dev		xusb_dev	usbd	XUSB reads
csw_xusb_devw	77	(0x4d)	xusb_dev		xusb_dev	usbd	XUSB_DEV writes

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csr_isprab	78	(0x4e)	isp2b		isp2b	ve2	ISP Read client for Crossbar A; instance b of isp2
csw_ispwab	80	(0x50)	isp2b		isp2b	ve2	ISP Write client for Crossbar A; instance b of isp2
csw_ispwbb	81	(0x51)	isp2b		isp2b	ve2	ISP Write client Crossbar B; instance b of isp2
csr_tsecsrd	84	(0x54)	tsec	tsec1	tsec	apb	TSEC Memory Return Data Client Description
csw_tsecswr	85	(0x55)	tsec	tsec1	tsec	apb	TSEC Memory Write Client Description
csr_a9avpscr	86	(0x56)	a9avp		a9avpsc	a9avppc	Reads from Cortex-A9 Shadow CPU core via the L2 cache
csw_a9avpscw	87	(0x57)	a9avp		a9avpsc	a9avppc	Writes from Cortex-A9 Shadow CPU core via the L2 cache
csr_gpusrd	88	(0x58)	gpu	gpub	gpu	gk	3D, ltcx reads
csw_gpuswr	89	(0x59)	gpu	gpub	gpu	gk	3D, ltcx writes
csr_displayt	90	(0x5a)	dc	dc1	display	dis	Display reads, Window T
csr_sdmmcra	96	(0x60)	sdmmc1a		sdmmca	sdm	sdmmca memory read client
csr_sdmmcraa	97	(0x61)	sdmmc2a		sdmmcaa	apb	sdmmcb memory read client
csr_sdmmcr	98	(0x62)	sdmmc3a		sdmmc	sd	sdmmc memory read client
csr_sdmmcrab	99	(0x63)	sdmmc4a		sdmmcab	apb	sdmmcd memory read client
csw_sdmmcwa	100	(0x64)	sdmmc1a		sdmmca	sdm	sdmmca memory write client
csw_sdmmcwaa	101	(0x65)	sdmmc2a		sdmmcaa	apb	sdmmcb memory write client
csw_sdmmcw	102	(0x66)	sdmmc3a		sdmmc	sd	sdmmc memory write client
csw_sdmmcwab	103	(0x67)	sdmmc4a		sdmmcab	apb	sdmmcd memory write client
csr_vicsrd	108	(0x6c)	vic		vic	vicpc	
csw_vicswr	109	(0x6d)	vic		vic	vicpc	
csw_viw	114	(0x72)	vi		vi2	ve	VI Write client
csr_displayd	115	(0x73)	dc		display	dis	Display reads, Window D

Client	ID	ID (hex)	SWGROUP	INBAND-SWID	Module	Partition	Description
csr_nvdecsrd	120	(0x78)	nvdec	nvdec1	nvdec	jpg	
csw_nvdecswr	121	(0x79)	nvdec	nvdec1	nvdec	jpg	
csr_aper	122	(0x7a)	ape		ape	aud	Audio Processing (APE) engine reads
csw_apew	123	(0x7b)	ape		ape	aud	Audio Processing (APE) engine writes
csr_nvjpgsrd	126	(0x7e)	nvjpg		nvjpg	jpg	
csw_nvjpgswr	127	(0x7f)	nvjpg		nvjpg	jpg	
csr_sesrd	128	(0x80)	se	se1	se	ahb	SE Memory Return Data Client Description
csw_seswr	129	(0x81)	se	se1	se	ahb	SE Memory Write Client Description
csr_axiapr	130	(0x82)	axiap		axiap	dfd	AXIAP reads
csw_axiapw	131	(0x83)	axiap		axiap	dfd	AXIAP writes
csr_etrr	132	(0x84)	etr		etr	dfd	ETR reads
csw_etrw	133	(0x85)	etr		etr	dfd	ETR writes
csr_tsecsrdb	134	(0x86)	tsecb	tsecb1	tsecb	apb	TSEC Memory Return Data Client Description; instance b of tsec
csw_tsecswrb	135	(0x87)	tsecb	tsecb1	tsecb	apb	TSEC Memory Write Client Description; instance b of tsec
csr_gpusrd2	136	(0x88)	gpu	gpub	gpu2	gk2	3D, ltcx reads; instance 2 of gpu
csw_gpuswr2	137	(0x89)	gpu	gpub	gpu2	gk2	3D, ltcx writes; instance 2 of gpu

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXXX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	X	ERR_TYPE: Type of the corresponding error: 0 = RSVD : Code 0x0 is never used. 2 = DECERR_EMEM : There is no physical DRAM attached for this address based on the value of EMEM_SIZE_MB configuration register. 3 = SECURITY_TRUSTZONE : The non-secure client has attempted to access the Trustzone-secured aperture; or a secure client has attempted to access the Trustzone-secured aperture using a non-secure request. 4 = SECURITY_CARVEOUT : Reserved 6 = INVALID_SMMU_PAGE : The SMMU page-translation flagged an error for that access.
27	X	ERR_INVALID_SMMU_PAGE_READABLE: If INVALID_SMMU_PAGE error, set if page permission was readable (AND of PTB/PDE/PTE). Ignore if no INVALID_SMMU_PAGE error.
26	X	ERR_INVALID_SMMU_PAGE_WRITABLE: If INVALID_SMMU_PAGE error, set if page permission was writable (AND of PTB/PDE/PTE). Ignore if no INVALID_SMMU_PAGE error.
25	X	ERR_INVALID_SMMU_PAGE_NONSECURE: If INVALID_SMMU_PAGE error, set if page permission was non-secure (AND of PTB/PDE/PTE). Ignore if no INVALID_SMMU_PAGE error.

Bit	Reset	Description
21:20	X	ERR_ADR_HI: Higher address bits of erring address whose lower bits are available in ERR_ADR register
18	X	ERR_SWAP: Swap bit for transaction
17	X	ERR_SECURITY: Set if the transaction was secure. 0 = NONSECURE 1 = SECURE
16	X	ERR_RW: Direction of the access that caused the error. 0 = READ 1 = WRITE
14:12	X	ERR_ADR1: Second subpartition unique address bits
7:0	X	ERR_ID: Client identifier (see above list) of the access that caused the error.

18.11.1.4 MC_ERR_ADR_0

Error Data Capture: Address

If a translation error is detected, this register records the request byte address.

Subsequent faults (of any type) will not change the status and address registers until the corresponding interrupt is cleared.

Note that the decode-error, security-Trustzone, and security-carveout checks occur on the physical address (in other words, after SMMU translation); thus those error types capture the physical address. All other errors capture the virtual address.

Offset: 0xc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ERR_ADR

18.11.1.5 MC_SMMU_CONFIG_0

Used for interrupt set/clear enums. When disabled, all transactions are untranslated. When enabled, transactions may be translated. See per-client and per-module enables in SMMU_* registers below.

This register can only be accessed by TrustZone-Secured accesses from the CPU.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

SMMU Enable Register

Offset: 0x10 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	SMMU_ENABLE: 0 = DISABLE 1 = ENABLE

18.11.1.6 MC_SMMU_TLB_CONFIG_0

Translation Lookaside Buffer Config Register

Controls usage of the TLB.

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

Offset: 0x14 | Read/Write: R/W | Reset: 0x30000030 (0b0011xxxxxxxxxxxxxxxxxxxx110000)

Bit	Reset	Description
31	DISABLE	TLB_STATS_ENABLE: Enable TLB Hit and Miss counters 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	DISABLE	TLB_STATS_TEST: Set stats registers to all "1s" 0 = DISABLE 1 = ENABLE
29	ENABLE	TLB_HIT_UNDER_MISS: Allow hits to pass misses in the TLB. This value may not be changed on the fly. Ideally, this should be set before enabling the SMMU. At the very least, the TLB needs to be flushed and traffic through the SMMU stopped before changing this value. 0 = DISABLE 1 = ENABLE
28	ENABLE	TLB_ROUND_ROBIN_ARBITRATION: When enabled, forces round robin (RR) arbitration between TLB hit and miss FIFOs. 0 = DISABLE 1 = ENABLE
5:0	0x30	TLB_ACTIVE_LINES: Set the number of active lines. Allows the TLB to be made "virtually smaller" to save power. "Inactive" lines will never hit and never hold data.

18.11.1.7 MC_SMMU_PTC_CONFIG_0

Controls usage of the PTC

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

Page Table Cache Config Register

Offset: 0x18 | Read/Write: R/W | Reset: 0x28000f3f (0b001x1000xxxxxxxxxxxx1111x0111111)

Bit	Reset	Description
31	DISABLE	PTC_STATS_ENABLE: Enable PTC Hit and Miss counters 0 = DISABLE 1 = ENABLE
30	DISABLE	PTC_STATS_TEST: Set stats registers to all "1s" 0 = DISABLE 1 = ENABLE
29	ENABLE	PTC_CACHE_ENABLE: Enable the PTC cache. 0 = DISABLE 1 = ENABLE
27:24	0x8	PTC_REQ_LIMIT: Limit outstanding PTC fill requests to the DRAM.
11:8	0xf	PTC_LINE_MASK: XOR pattern for line generation
6:0	0x3f	PTC_INDEX_MAP: XOR pattern for tag generation

18.11.1.8 MC_SMMU_PTB_ASID_0

Page Table Bases (PTBs)

These pointers point to a 4KB-page aligned table of PDEs for each ASID. They also contain initial permission bits that are ANDed with those in the PDE and PTEs to determine the final permissions of accesses through this ASID.

The PTBs are stored in a RAM that is accessed indirectly though the SMMU_PTB_ASID and SMMU_PTB_DATA registers.

Boot requirements: This RAM should be saved to SDRAM and restored by the OS during warm boot.

Page Table Base ASID Register

This register selects which PTB is modified by the SMMU_PTB_DATA register.

Offset: 0x1c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:0	X	CURRENT_ASID: ASID used to address SMMU_PTB register

18.11.1.9 MC_SMMU_PTB_DATA_0

Writes or reads to this register write or read the Page Table Base pointer for the ASID defined in the SMMU_PTB_ASID register.

This register is not TrustZone secure, but only a TrustZone secure access may alter the PTB for a secure ASID.

Secure ASIDs have additional privilege over other non-secure ASIDs - they may fetch page tables from secure memory and they can allow clients translated through them to make secure accesses if programmed to do so.

Attempts to write a secure ASID with a non-secure access will be ignored, and non-secure reads will return garbage.

Page Table Base Data Register

Offset: 0x20 | Read/Write: R/W | Reset: 0xX0XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	ASID_READABLE: if set, reads are allowed for this ASID
30	X	ASID_WRITABLE: if set, writes are allowed for this ASID
29	X	ASID_NONSECURE: if set, non-secure accesses are allowed for this ASID
21:0	X	ASID_PDE_BASE: Pointer to page of PDEs, bits [33:12] for this ASID

18.11.1.10 MC_SMMU_TLB_FLUSH_0

Software can write this register to flush a specific page group, 4 MB section, entire ASID or the entire TLB. There is independent control over matching part or all of the VA and/or matching the ASID.

Note: *The granularity of VA mapping here is at a page group, which is the number of PTEs that fit in a memory atom.*

Translation Lookaside Buffer Flush Register

Offset: 0x30 | Read/Write: R/W | Reset: 0xXX0XXXXX (0bxxxxxxxxxxxx0xxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TLB_FLUSH_ASID_MATCH: If enabled, only entries matching TLB_FLUSH_ASID are flushed. 0 = DISABLE 1 = ENABLE
30:24	X	TLB_FLUSH_ASID: ASID to match when TLB_FLUSH_ASID_MATCH is ENABLED
19:2	X	TLB_FLUSH_VA_GROUP: Virtual address to match for group flushes, VA[31:15]. VA[14] of this field is ignored due to the TLB line size.
1:0	X	TLB_FLUSH_VA_MATCH: Controls flushing based on VA, one of ALL: Flush entire TLB section: Flush all entries with a VA[31:22] that match TLB_FLUSH_VA_. GROUP: Flush all entries with a VA[31:14] that match TLB_FLUSH_VA_GROUP. By default, the GROUP setting takes effect if this field is programmed to a value of 1. 0 = ALL 2 = SECTION 3 = GROUP

18.11.1.11 MC_SMMU_PTC_FLUSH_0

Software can use this register to flush a specific PTE or PDE from the Page Table Cache (PTC) by writing the atom-aligned physical address of the PTE group. It can also flush the entire PTC via this register.

Note: *The granularity of flushing is at the PTE group, which is the number of PTEs that fit in a memory atom.*

Page Table Cache Flush Register

Offset: 0x34 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx)

Bit	Reset	Description
31:4	X	PTC_FLUSH_ADR: Physical address of PTE group to match for address flushes, PA[31:4]
3	0x1	PTC_NO_WAIT_IDLE_FLUSH: Set bit to immediately flush without waiting for hit FIFO to go idle. Not required because it does not present an ordering hazard.
0	X	PTC_FLUSH_TYPE: Controls flushing, one of ALL: Flush entire PTC ADR: Flush PTEs that are addressed by PTC_FLUSH_ADR 0 = ALL 1 = ADR

18.11.1.12 MC_SMMU_ASID_SECURITY_0

ASID Security Register This register controls which ASIDs are considered "secure". Secure ASIDs cannot have their PTBs read or written by a non-secure access, nor can they be assigned as a module's ASID by a non-secure write. ASIDs may also be enabled as "promoting", which allows modules mapped through them to inherit their secure status. If a secure ASID has its promotion bit enabled, modules mapped to this ASID may access the secure physical memory region.

> ***Note:*** *All ASIDs, regardless of security, may access TrustZone-secure physical memory for page table accesses. This register can only be accessed by TrustZone-Secured accesses from the CPU. Registers for ASID 0 through 31.*

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x38 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS: If bit N in this field is set, clients that are mapped to ASID N may access the TrustZone-secure area of memory if ASID N is also marked as "secure".
15:0	0x0	SECURE_ASIDS: If bit N in this field is set, the PTB of ASID N cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.13 MC_SMMU_ASID_SECURITY_1_0

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS_1: If bit N in this field is set, clients that are mapped to ASID N+16 may access the TrustZone-secure area of memory if ASID N+16 is also marked as "secure".
15:0	0x0	SECURE_ASIDS_1: If bit N in this field is set, the PTB of ASID N+16 cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.14 MC_SMMU_ASID_SECURITY_2_0

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x9e0 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS_2: If bit N in this field is set, clients that are mapped to ASID N+32 may access the TrustZone-secure area of memory if ASID N+32 is also marked as "secure".
15:0	0x0	SECURE_ASIDS_2: If bit N in this field is set, the PTB of ASID N+32 cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.15 MC_SMMU_ASID_SECURITY_3_0

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x9e4 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS_3: If bit N in this field is set, clients that are mapped to ASID N+48 may access the TrustZone-secure area of memory if ASID N+48 is also marked as "secure".
15:0	0x0	SECURE_ASIDS_3: If bit N in this field is set, the PTB of ASID N+48 cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.16 MC_SMMU_ASID_SECURITY_4_0

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x9e8 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS_4: If bit N in this field is set, clients that are mapped to ASID N+64 may access the TrustZone-secure area of memory if ASID N+64 is also marked as "secure".
15:0	0x0	SECURE_ASIDS_4: If bit N in this field is set, the PTB of ASID N+64 cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.17 MC_SMMU_ASID_SECURITY_5_0

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x9ec | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS_5: If bit N in this field is set, clients that are mapped to ASID N+80 may access the TrustZone-secure area of memory if ASID N+80 is also marked as "secure".
15:0	0x0	SECURE_ASIDS_5: If bit N in this field is set, the PTB of ASID N+80 cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.18 MC_SMMU_ASID_SECURITY_6_0

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x9f0 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS_6: If bit N in this field is set, clients that are mapped to ASID N+96 may access the TrustZone-secure area of memory if ASID N+96 is also marked as "secure".
15:0	0x0	SECURE_ASIDS_6: If bit N in this field is set, the PTB of ASID N+96 cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.19 MC_SMMU_ASID_SECURITY_7_0

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x9f4 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PROMOTING_ASIDS_7: If bit N in this field is set, clients that are mapped to ASID N+112 may access the TrustZone-secure area of memory if ASID N+112 is also marked as "secure".

Bit	Reset	Description
15:0	0x0	SECURE_ASIDS_7: If bit N in this field is set, the PTB of ASID N+112 cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

18.11.1.20 MC_EMEM_CFG_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- During warm boot, this register may be derived from EMEM_ADR_CFG* register values.

External Memory Aperture Configuration

Offset: 0x50 | Read/Write: R/W | Reset: 0x00002000 (0b0xxxxxxxxxxxxxxxx10000000000000) | Default: 0x00000000

Bit	Reset	Software Default	Description
31	B2GB	B2GB	EMEM_BOM: Base of DRAM memory in GBs [PMC_SECURE]. B2GB should be used for 34b systems, while B0GB supports up to 16GB using extended mapping mode 1 = B0GB 0 = B2GB
13:0	0x7ff	NONE	EMEM_SIZE_MB: Specify the external memory aperture size in megabytes. This field must be set to less than or equal to the available physical memory for proper operation. This value is used to check for decode errors; requests to addresses greater than or equal to EMEM_BOM+EMEM_SIZE_MB will result in an EMEM decode error.

18.11.1.21 MC_EMEM_ADR_CFG_0

The EMEM_ADR_CFG* registers are used to specify the DRAM density and geometry parameters. The MC will use these parameters to derive device, row, bank, column addressing values to EMC.

> ***Note:*** *The maximum size externally supported is dependent on pinout and address-map aperture limitations, and that it may be possible to configure the device/row/bank/column addresses in ways that exceed these limitations.*

For each device attached [1..NUMDEV], the associated per-device address configuration must also be programmed, in the associated register EMEM_ADR_CFG_DEV{N-1}.

The maximum number of row bits carried internally is defined by NV_MC_EMEM_ROW_WIDTH.

The maximum number of address pins is defined by NV_MC_EMEM_ROWPIN_WIDTH.

If not being used to address a device, the chip-select pins may be re-used as row address pins.

Valid settings for EMEM_ADR_CFG_DEV* registers should ensure device address width <= bank width + column width + NV_MC_EMEM_ROW_WIDTH

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

External Memory Address Configuration System

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	N1	EMEM_NUMDEV: [PMC] number of populated DRAM devices. 0 = N1 1 = N2

18.11.1.22 MC_EMEM_ADR_CFG_DEV0_0

Configures the density and geometry of the first attached device.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Note: *DEVSIZE is per subpartition.*

External Memory Address Configuration Device 0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00040202 (0bxxxxxxxxxxxx0100xxxxxx10xxxxx010)

Bit	Reset	Description
19:16	D64MB	EMEM_DEV0_DEVSIZE: [PMC] density of the first attached DRAM device, used to generate width of row address. 2 = D16MB 3 = D32MB 4 = D64MB 5 = D128MB 6 = D256MB 7 = D512MB 8 = D1024MB 8 = D1GB 9 = D2048MB 9 = D2GB 10 = D4096MB 11 = D8192MB 12 = D768MB 13 = D384MB
9:8	W2	EMEM_DEV0_BANKWIDTH: [PMC] width of bank address of the first attached DRAM device. 2 = W2 3 = W3
2:0	W9	EMEM_DEV0_COLWIDTH: [PMC] width of column address of the first attached DRAM device. W8 maps to 1KB page size, W9 to 2KB, W10 to 4KB, etc. 1 = W8 2 = W9 3 = W10 4 = W11 5 = W12

18.11.1.23 MC_EMEM_ADR_CFG_DEV1_0

Configures the density and geometry of the second attached device, if a second device is populated.

To allow for 2 devices with different column width/bank width to be used without creating holes in the system-level address map, the DEVSIZE setting of the second device may be smaller than the others to allow for non-powers-of-2 attached memory sizes.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Note: *DEVSIZE is per subpartition.*

External Memory Address Configuration Device 1

Offset: 0x5c | Read/Write: R/W | Reset: 0x00040202 (0bxxxxxxxxxxxx0100xxxxxx10xxxxx010)

Bit	Reset	Description
19:16	D64MB	EMEM_DEV1_DEVSIZE: [PMC] Density of the second attached DRAM device, used to generate width of row address. 2 = D16MB 3 = D32MB 4 = D64MB 5 = D128MB 6 = D256MB 7 = D512MB 8 = D1024MB 8 = D1GB 9 = D2048MB 9 = D2GB 10 = D4096MB 11 = D8192MB 12 = D768MB 13 = D384MB
9:8	W2	EMEM_DEV1_BANKWIDTH: [PMC] width of bank address of the second attached DRAM device. 2 = W2 3 = W3
2:0	W9	EMEM_DEV1_COLWIDTH: [PMC] width of column address of the second attached DRAM device. W8 maps to 1KB page size, W9 to 2KB, W10 to 4KB, etc. 1 = W8 2 = W9 3 = W10 4 = W11 5 = W12

18.11.1.24 MC_EMEM_ADR_CFG_CHANNEL_MASK_0

External Memory Address Configuration Channel Select Mask

Configures the routing of requests between the two memory channels. The channel select mask is ANDed with the address of a transaction. The resulting value is XORed to a single bit, which is used to select the channel for the transaction.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Mask bits [10:9] select single channel vs. dual-channel modes and 512B vs. 1KB interleave as follows, each with its own physical address to mapping.

Interleave	MASK[10:9]	Remarks
Single channel	2’b00	
512B	2’bx1	Channels alternate on (most) 512B boundaries
1KB	2’b10	Channels alternate on (most) 1KB boundaries while decoding the DRBC data from the address, if mask bit 9th is set then the 9th bit of address is dropped for DRBC decoding and else if mask bit 10 is set then 10th bit address is dropped for address decoding.

Write access to this register is controlled by the EMEM_CFG_ACCESS_CTRL_0 register.

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000xxxxxxxxx)

Bit	Reset	Description
31:9	0x0	EMEM_CHANNEL_MASK: [PMC_SECURE] Mask is ANDed with address and the resulting value is XORed to a single bit

18.11.1.25 MC_EMEM_ADR_CFG_BANK_MASK_0_0

External Memory Address Configuration Bank Select Mask

Configures the calculation of the DRAM bank. The Bank select mask N is ANDed with the address of a transaction. The resulting value is XORd to a single bit, which forms the Nth bit of the bank.

Note: In dual channel mode, MC drops the channel bit (bit 9 or bit 10 depending on the channel mask) and applies the bank mask on the right-shifted address.

- Single channel mode: bank[0] = ^(bank_mask_0[31:10] & address[31:10]);
- Dual channel mode: bank[0] = ^(bank_mask_0[31:10] & address[32:11]);

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Write access to these registers is controlled by the EMEM_CFG_ACCESS_CTRL register

Offset: 0x64 | Read/Write: R/W | Reset: 0xeb772400 (0b11101011011101110011001xxxxxxxxxx)

Bit	Reset	Description
31:10	0x3addc9	EMEM_BANK_MASK_0: [PMC_SECURE] Mask is ANDed with address and the resulting value is XORd to form bank[0]

18.11.1.26 MC_EMEM_ADR_CFG_BANK_MASK_1_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x249e4800 (0b0010010010011110010010xxxxxxxxxx)

Bit	Reset	Description
31:10	0x92792	EMEM_BANK_MASK_1: [PMC_SECURE] Mask is ANDed with address and the resulting value is XORd to form bank[1]

18.11.1.27 MC_EMEM_ADR_CFG_BANK_MASK_2_0

Offset: 0x6c | Read/Write: R/W | Reset: 0x9dbbb000 (0b1001110110111011101100xxxxxxxxxx)

Bit	Reset	Description
31:10	0x276eec	EMEM_BANK_MASK_2: [PMC_SECURE] Mask is ANDed with address and the resulting value is XORed to form bank[2]

18.11.1.28 MC_SECURITY_CFG0_0

Security aperture: Can be only accessed by TrustZone-Secured accesses from the secure clients. Access to this register is restricted to TrustZone-secured requestors.

Boot requirements: This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

The global memory client IDs with TrustZone-level security are as follows:

CLIENT	ID	ID (hex)	SWGROUP	MODULE	PARTITION	DESCRIPTION
csr_ptcr	0	(0x00)	ptc			Misses from System Memory Management Unit (SMMU) Page Table Cache (PTC)

CLIENT	ID	ID (hex)	SWGROUP	MODULE	PARTITION	DESCRIPTION
csr_host1xdmar	22	(0x16)	hc	host1x	host	Host channel data reads
csr_ppcsahbslvr	30	(0x1e)	ppcs	ppcs	ahb	AHB bus reads
csr_mpcorer	39	(0x27)	cpu	cpu	ftop	Reads from 4 CPU cores via the L2 cache
csw_mpcorew	57	(0x39)	mpcore	mpcore	ftop	Writes from 4 CPU cores via the L2 cache
csw_ppcsahbslvw	60	(0x3c)	ppcs	ppcs	ahb	AHB bus writes
csr_a9avpscr	86	(0x56)	a9avp	a9avpsc	a9avppc	Reads from Cortex-A9 Shadow CPU core via the L2 cache
csw_a9avpscw	87	(0x57)	a9avp	a9avpsc	a9avppc	Writes from Cortex-A9 Shadow CPU core via the L2 cache
csr_displayt	90	(0x5a)	dc	display	dis	Display reads, Window T
csr_sesrd	128	(0x80)	se	se	ahb	SE Memory Return Data Client Description
csw_seswr	129	(0x81)	se	se	ahb	SE Memory Write Client Description
csr_axiapr	130	(0x82)	axiap	axiap	dfd	AXIAP reads
csw_axiapw	131	(0x83)	axiap	axiap	dfd	AXIAP writes
csr_etrr	132	(0x84)	etr	etr	dfd	ETR reads
csw_etrw	133	(0x85)	etr	etr	dfd	ETR writes

Secure Region Configuration: Base

Offset: 0x70 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x80000000 (0b100000000000xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:20	0x800	SECURITY_BOM: SECURITY_BOM is the base of the secured region, limited to MB granularity. This must point to a region of the physical address map allocated to EMEM for it to be effective; the MC cannot secure address space it does not own. (In other words, this is an absolute address, not an offset.) Above is the list of clients with the TrustZone-security access. Note that AXICIF clients will adhere to the standard AXI protocol "aprot[1]==0" indication for secure requests.

18.11.1.29 MC_SECURITY_CFG1_0

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

Secure Region Configuration: Bound

Offset: 0x74 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	SECURITY_SIZE_MB: SECURITY_SIZE_MB is the size, in megabytes, of the secured region. If set to 0, the security check in MC is disabled.

18.11.1.30 MC_EMEM_ARB_CFG_0

General configuration of the External Memory Arbiter.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.

- During Boot ROM warm boot, this register may be derived from the emcclk and a standard tick-value.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000008 (0b00000000xxx00000xxxxxxx000001000)

Bit	Reset	Description
20:16	0x0	EXTRA_TICKS_PER_UPDATE: The number of extra ticks to add every deadline timer update. Used to keep cycles-per-tick constant when the mcclk period is less than the chosen wall-clock granularity for the deadline counter. If =0, then every tick increments the deadline counter by 1; if =1, then every tick increments the deadline counter by 2, etc. Assuming a target of 30ns-per-tick, some example programming (assumes DIV=2): 10MHz emcclk => 5MHz mcclk => would need to increment deadline counter by 6.67 if using 1 cycle-per-tick => or update by 20 every 3 cycles => 3 cycles-per-tick and 19 extra ticks per update. Since performance is not critical at such slow speeds, rounding errors may be deemed acceptable, so the previous example may be simpler to program as 1 cycle-per-tick and 5 extra ticks per update. Another example, the slowest supported clock speed using a 30ns tick that doesn't distort the tick is: 1.04MHz mcclk => with 31 extra ticks per update and updating every cycle.
8:0	0x8	CYCLES_PER_UPDATE: The number of mcclk cycles per deadline timer update. The target wall-clock granularity ("tick") for the deadline counter should be fixed for the design, then the CYCLES_PER_UPDATE should be used to convert the wall-time value into mcclk cycles. All client latency allowances are also expressed in units of "ticks". Of all deadline-related controls, only the CYCLES_PER_UPDATE and EXTRA_TICKS_PER_UPDATE values need to be updated on a clock-change event. Assuming a target of 30ns per tick, some example programming (assumes DIV=2): 667MHz emcclk => 333MHz mcclk => 10 cycles per tick; 400MHz emcclk => 200MHz mcclk => 6 cycles-per-tick. Note that value 0x0 is illegal.

18.11.1.31 MC_EMEM_ARB_OUTSTANDING_REQ_0

This register can be used to limit the number of outstanding requests in the arbiter and monitor the count.

If outstanding transaction is greater than the maximum, requests are throttled based on MC_EMEM_ARB_RING1_THROTTLE_0 register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from other MC settings.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: Outstanding Request Limiter

Offset: 0x94 | Read/Write: R/W | Reset: 0x8XXX0040 (0b10xxxxxxxxxxxxxxxxxxxxx001000000)

Bit	R/W	Reset	Description
31	RW	ENABLE	LIMIT_OUTSTANDING: when ENABLED, the total number of requests in the arbiter is limited to the value in ARB_MAX_OUTSTANDING 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	LIMIT_DURING_HOLDOFF_OVERRIDE: when DISABLED, override the limiting of transactions during hold-off to let requests flow freely 0 = DISABLE 1 = ENABLE
24:16	RO	X	ARB_OUTSTANDING_COUNT: Current number of outstanding requests
8:0	RW	0x40	ARB_MAX_OUTSTANDING: The maximum number of requests allowed in the arbiter when LIMIT_OUTSTANDING is set to ENABLE.

18.11.1.32 MC_EMEM_ARB_TIMING_RCD_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tRCD

Offset: 0x98 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RCD: The minimum number of cycles between activate commands to the same bank. Program to maximum (1,ceil(max(EMC.WR_RCD,EMC.RD_RCD)/DIV)-1-1). Note that value 0x0 is illegal.

18.11.1.33 MC_EMEM_ARB_TIMING_RP_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tRP

Offset: 0x9c | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	RP: The minimum number of cycles between an internal precharge command and an activate command to the same bank. Program to ceil (EMC.RP/DIV)-1+SFA. The combined value of RAP2PRE+RP 0x0 is illegal; the combined value of WAP2PRE+RP 0x0 is illegal.

18.11.1.34 MC_EMEM_ARB_TIMING_RC_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing : tRC

Offset: 0xa0 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	RC: This is the minimum number of cycles between activate commands to the same bank. Program to ceil (max(EMC.RC,(EMC.RAS+EMC.RP))/DIV)-1.

18.11.1.35 MC_EMEM_ARB_TIMING_RAS_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tRAS

Offset: 0xa4 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	RAS: This is the minimum number of cycles between an activate command and a precharge command to the same bank. Program to ceil (EMC.RAS/DIV)-1-1.

18.11.1.36 MC_EMEM_ARB_TIMING_FAW_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tFAW

Offset: 0xa8 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
5:0	0x3f	FAW: For 8-bank devices: Only 4 activates may occur within this rolling window. Program to ceil (EMC.TFAW/DIV)-1. Programming this to 0x0 or not programming a device to 4 banks will turn off this check.

18.11.1.37 MC_EMEM_ARB_TIMING_RRD_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tRRD

Offset: 0xac | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	RRD: The minimum number of cycles between an activate command and an activate command to a different bank. Program to ceil (EMC.RRD/DIV)-1.

18.11.1.38 MC_EMEM_ARB_TIMING_RAP2PRE_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tRAP2PRE

Offset: 0xb0 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	RAP2PRE: The number of cycles between a read command with auto-precharge and the internal precharge command generated by the DRAM. Program to ceil (EMC_R2P/DIV). Note that: the combined value of RAP2PRE+RP 0x0 is illegal.

18.11.1.39 MC_EMEM_ARB_TIMING_WAP2PRE_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tWAP2PRE

Offset: 0xb4 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	WAP2PRE: The number of cycles between a write command with auto-precharge and the internal precharge command generated by the DRAM. Program to ceil (EMC_W2P/DIV). Note that: the combined value of WAP2PRE+RP 0x0 is illegal.

18.11.1.40 MC_EMEM_ARB_TIMING_R2R_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tR2R

Offset: 0xb8 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	R2R: The number of cycles between consecutive read commands to different devices (different chip selects). Program to ceil (EMC.REXT/DIV)-1+OTFA+SFA.

18.11.1.41 MC_EMEM_ARB_TIMING_W2W_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tW2W

Offset: 0xbc | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	W2W: The number of cycles between consecutive write commands to different devices (different chip selects). Program to ceil (EMC.WEXT/DIV)-1+SFA.

18.11.1.42 MC_EMEM_ARB_TIMING_R2W_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tR2W

Offset: 0xc0 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	R2W: The number of cycles to turn the bus from reads to writes. Program to ceil (EMC.R2W/DIV)-1+OTFA+SFA.

18.11.1.43 MC_EMEM_ARB_TIMING_W2R_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: DRAM Timing: tW2R

Offset: 0xc4 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	W2R: The number of cycles to turn the bus from reads to writes. Program to ceil (EMC.W2R/DIV)-1+SFA.

18.11.1.44 MC_EMEM_ARB_DA_TURNS_0

The direction arbiter attempts to choose the highest efficiency (i.e., lowest bubbles) direction to open pages next. While programming this register incorrectly may lower performance, it should never cause chip-hangs since the EMC always controls the final DRAM timing decisions. This register should be reprogrammed after clock-change events to bring the computed values back in alignment with EMC's register values. These turn costs are deliberately not computed in hardware to provide better flexibility to tweak arbiter performance.

The configuration registers for turns for DA are separate from the timing registers to allow for flexibility in the programming.

Boot requirements:

- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from MC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: Direction Arbiter: Turns

Offset: 0xd0 | Read/Write: R/W | Reset: 0x0f0f0205 (0b00001111000011110000001000000101)

Bit	Reset	Description
31:24	0xf	W2R_TURN: Bubbles produced by a write to read bus turn. Approximately MC_EMEM_ARB_TIMING_W2R.
23:16	0xf	R2W_TURN: Bubbles produced by a read to write bus turn. Approximately MC_EMEM_ARB_TIMING_R2W.
15:8	0x2	W2W_TURN: Bubbles produced by a write to write (other device) bus turn. Approximately MC_EMEM_ARB_TIMING_W2W.
7:0	0x5	R2R_TURN: Bubbles produced by a read to read (other device) bus turn. Approximately MC_EMEM_ARB_TIMING_R2R.

18.11.1.45 MC_EMEM_ARB_DA_COVERS_0

The direction arbiter attempts to choose the highest efficiency (i.e., lowest bubbles) direction to open pages next. While programming this register incorrectly may lower performance, it should never cause chip-hangs since the EMC always controls the final DRAM timing decisions. This register should be reprogrammed after clock-change events to bring the computed values back in alignment with EMC's register values. These cover costs are deliberately not computed in hardware to provide better flexibility to tweak arbiter performance.

Boot requirements:

- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from MC settings using the given equations.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: Direction Arbiter: Covers

Offset: 0xd4 | Read/Write: R/W | Reset: 0x000f0f0f (0bxxxxxxxx00001111000011110000 1111)

Bit	Reset	Description
23:16	0xf	RCD_W_COVER: Cost to cover the activate-to-write delay. Approximately ceil ((EMC.RTP + EMC.RP + EMC.RD_RCD)/DIV)-1.
15:8	0xf	RCD_R_COVER: Cost to cover the activate-to-read delay. Approximately ceil ((EMC.WTP + EMC.RP + EMC.WR_RCD)/DIV)-1.
7:0	0xf	RC_COVER: Cost to cover the activate-to-activate delay. Approximately MC_EMEM_ARB_TIMING_RC.

18.11.1.46 MC_EMEM_ARB_MISC0_0

BC2AA_HOLDOFF: In the case where the open pages have more work pending than the time it would take to open another page, it is better to hold off opening that page until it is truly needed. The Best-bank Cache computes the pending work and compares it to this threshold and advises the Activation Arbiter to hold-off.

Priority Inversion: In general, if the arbiter is working on a lower-priority request that blocks a higher-priority request, these thresholds are used to limit the number of low-priority requests serviced before the arbiter interrupts the lower-priority traffic to start servicing the higher-priority traffic.

The priority classes are (in increasing priority order): Low (! expired), High (expired), HighIso.

There are three ways priority can be inverted: Bank Activation, Bus Direction, and Refresh:

Bank Activation: If an unexpired request to bank A, row R holds the page open, and a request to bank A, row S expires, the Transaction Arbiter must make a decision whether to finish out the work for row R or to interrupt that transfer to immediately service row S. The TA makes this decision based on the remaining work for row R and whether row S is for an isochronous client, and the two PRIORITY_INVERSION thresholds.

Bus Direction: If the Transaction Arbiter is working on expired requests in direction P, and requests are also expired (or expired-iso) in direction Q, the Transaction Arbiter must make a decision whether to continue working in direction P or switch the bus to direction Q. The Transaction Arbiter will service at maximum PRIORITY_INVERSION_THRESHOLD (or _INVERSION_ISO_THRESHOLD) requests in direction P before switching to direction Q.

Refresh: If the Transaction Arbiter is working on requests, and EMC asks to inject a refresh, the Transaction Arbiter must decide whether it is more efficient (overall) to close the pages immediately or to finish the pages and then close them. Depending on the setting of the REFRESH_ACK_THRESHOLD_USAGE configuration bit (in EMEM_ARB_MISC1 register, below), the Transaction Arbiter will either choose to close the page immediately (NONE), close the page immediately if its length is over the iso threshold (ISO), or close the page immediately if its length is over the normal threshold (NORMAL). If the length is less than the chosen threshold, the Transaction Arbiter will service all remaining requests to the page before acknowledging the EMC refresh request.

When in MC_EMC_SAME_FREQ mode, the priority inversion thresholds should be set to half of their non-same-frequency values.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- The MC_EMC_SAME_FREQ field should be saved to scratch registers and restored by the Boot ROM during warm boot.
- The other fields of this register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from other settings.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: Miscellaneous Thresholds (0)

Offset: 0xd8 | Read/Write: R/W | Reset: 0x720a4010 (0bx11100100000101001000000000010000)

Bit	Reset	Description
30:28	0x7	ATOMS_PER_DVFS_PULSE: [PMC] ATOMS_PER_DVFS_PULSE is used to control the rate at which the MC updates the sys_stats_mon outputs. The MC will toggle the sys_stats_mon outputs for every 2^(ATOMS_PER_DVFS_PULSE+1) atoms of traffic, where an atom is defined as any transaction. Note that 16B, 32B, and 64B requests consume the same amount of memory bus resources. So, in this context, an atom is defined as any request and not cumulative 64B sized transactions. The maximum ATOMS_PER_DVFS_PULSEsetting supported is the reset value, also known as NV_MC_EMEM_DVFS_CNTR_CFG_RESET.
27	DISABLE	MC_EMC_SAME_FREQ: [PMC] Used in conjunction with EMEM_ARB_MISC1.COALESCE_FOR_PERFORMANCE to configure how often the MC can send EMC a data request, should be configured to mirror CAR's CLK_SOURCE_EMC.MC_EMC_SAME_FREQ. If configured to DIV=2 BL=4, then the MC can send a data request every cycle. If configured to DIV=1 BL=4, then the MC can send a data request every other cycle. If configured to DIV=2 BL!=4, then MC can send a data request every other cycle. If configured to DIV=1 BL!=4, then the MC can send a data request once every four cycles. 0 = MC is 1/2 the frequency of EMC (DIV=2). 1 = MC is the same frequency of EMC (DIV=1). 0 = DISABLE 1 = ENABLE
26:21	0x10	EXPIRING_SOON_SLACK_THRESHOLD: Slack threshold below which an isochronous request is considered to be expiring "soon". Used to de-assert mc2emc_idle signal early to wake EMC out of power-down or self-refresh. If using EMC's Dynamic Self Refresh features, set this to the ticks needed to cover exit self-refresh delay; otherwise, if using EMC's ACPD feature, set to the ticks needed to cover exit power-down delay; if neither power-saving feature is being used, the programming of this threshold should have no effect.
20:16	0xa	PRIORITY_INVERSION_ISO_THRESHOLD: Bank Activation: maximum number of unexpired or expired-but-not-isochronous requests that can be serviced before switching to expired isochronous traffic. Bus Direction: maximum number of requests in direction P that can be serviced before switching to expired-isochronous traffic in direction Q. Refresh: maximum number of requests that can be serviced before refresh is sent. Values <= 0x2 disable the feature. Values < 0x6 should be avoided if coalescing for performance.

Bit	Reset	Description
15	DISABLE	EMC_REQ_B2B_XFER: When enabled, the MC allows EMC transactions on back-to-back clocks regardless of EMC consumption bandwidth 0 = DISABLE 1 = ENABLE
14:8	0x40	PRIORITY_INVERSION_THRESHOLD: Bank Activation: maximum number of unexpired requests that can be serviced before switching to expired traffic. Bus Direction: maximum number of requests in direction P that can be serviced before switching to expired traffic in direction Q. Refresh: maximum number of requests that can be serviced before refresh is sent. Values <= 0x2 disable the feature. Values < 0x6 should be avoided if coalescing for performance.
7:0	0x10	BC2AA_HOLDOFF_THRESHOLD: If the pending work is greater than this value, hold off on activations. Generally should be set to match the page-opening cost: MC_EMEM_ARB_TIMING_RC+1.

18.11.1.47 MC_EMEM_ARB_MISC1_0

Two DVFS counters exist in the "NV_MC_ARB_EMEM_stats.v" module: one for all requests and another for CPU-only transactions. Both of these are down counters which decrement for each transaction atom on mcclk.

When these counters reach zero, a single mcclk pulse is generated, and the counters reinitialize to a value of ((2^(ATOMS_PER_DVFS_PULSE+1))-1). The single mcclk pulse that is generated (when the counters reach zero) gets synchronized to sclk (system clock) through a cross-clock domain FIFO control module, which has a minimum response time requirement of (4 * sclk_period + 4 * mcclk_period). Therefore, ATOMS_PER_DVFS_PULSE should never be set such that ((2^(ATOMS_PER_DVFS_PULSE+1)) * mcclk_period) is less than (4 * sclk_period + 4 * mcclk_period).

This translates to the following rule for setting the minimum value of ATOMS_PER_DVFS_PULSE:

ATOMS_PER_DVFS_PULSE >= log2((4 * sclk_period/mcclk_period) + 4) - 1

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- The COALESCE_FOR_PERFORMANCE field should be saved to scratch registers and restored by the Boot ROM during warm boot. It may also be derived from EMC settings.
- The other fields of this register should be saved to SDRAM and restored by the OS during warm boot.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

External Memory Arbitration Configuration: Miscellaneous Thresholds

Offset: 0xdc | Read/Write: R/W | Reset: 0x80a0000a (0b1000xx00101xxxxxxxx0000000001010) | Default: 0x00000008

Bit	Reset	Software Default	Description
31:28	0x8	NONE	DEADLOCK_PREVENTION_SLACK_THRESHOLD: If the slack at the head of any bank-queue, or any of the slacks in the hit-under-miss FIFO is less than -(2^threshold), backpressure the input to the arbiter until the violating slack values have been arbitrated. This is intended to: (a) avoid deadline-wrapping in heavily backpressured corner-cases and (b) quickly get such cases back into high-efficiency arbitration. Value of 0x0 disables the feature. The reset value is NV_MC_EMEM_DL_WIDTH-1. Behavior with values greater than or equal to NV_MC_EMEM_DL_WIDTH is UNDEFINED. If the threshold is reached, ARBITRATION_EMEM_INT will fire.
25:24	NORMAL	NONE	REFRESH_ACK_THRESHOLD_USAGE: Determines which thresholds (if any) the refresh-acknowledge signal will use to determine how to close all pages after EMC has requested a refresh. 0 = NORMAL : Use priority-inversion threshold 1 = ISO : Use priority-inversion-ISO threshold 2 = NONE : Force a precharge immediately to close the page

Bit	Reset	Software Default	Description
23:21	USE_EXPIRING_SOON_SLACK_THRESHOLD	NONE	EXPIRING_SOON_SLACK_THRESHOLD_PD: [PMC] Slack threshold below which an isochronous request is considered to be expiring "soon". Used to deassert the mc2emc_idle_pd signal early to wake the EMC out of power-down. (EXPIRING_SOON_SLACK_THRESHOLD controls exit from self-refresh.) This register should be set to the ticks needed to cover the exit power-down delay. If the available choices are not sufficient, the value should be set in EXPIRING_SOON_SLACK_THRESHOLD and this register should be set to USE_EXPIRING_SOON_SLACK_THRESHOLD 0 = ZERO_TICKS : 0 tick threshold for PD expiring soon 1 = ONE_TICK: 1 tick threshold for PD expiring soon 2 = TWO_TICKS: 2 tick threshold for PD expiring soon 3 = FOUR_TICKS: 4 tick threshold for PD expiring soon 4 = EIGHT_TICKS: 8 tick threshold for PD expiring soon 5 = USE_EXPIRING_SOON_SLACK_THRESHOLD: Use the setting in EXPIRING_SOON_SLACK_THRESHOLD to control mc2emc_idle_pd
12:4	0x0	0x0	ALT_DEADLOCK_PREVENTION_SLACK_THRESHOLD: [PMC] Use the absolute value of deadlock prevention slack threshold instead of 2^n format. Note that the DEADLOCK_PREVENTION_SLACK_THRESHOLD field still needs to be programmed along with this one (for example, 7).
3	ENABLE	ENABLE	COMBINED_INTERRUPT_MODE: [PMC] 1 = combined interrupt mode. 0 = independent interrupt mode. Shadowed 0 = DISABLE 1 = ENABLE
2	DISABLE	0x0	BLOCK_LP_CPU_WR_IF_SMMU_INP_HP: [PMC] When enabled, low priority (due to DDA) CPU writes get blocked when the SMMU output is high priority and a request is present at the input to the SMMU. 0 = DISABLE 1 = ENABLE
1	ENABLE	NONE	ALLOW_BCA_HOLDOFF_WHEN_EXP: [PMC] When enabled, allows BCA holdoff to occur even when expired requests are present. 0 = DISABLE 1 = ENABLE
0	DISABLE	0x0	BLOCK_LP_CPU_RD_IF_SMMU_INP_HP: [PMC] When enabled, low priority (due to DDA) CPU reads get blocked when the SMMU output is high priority and a request is present at the input to the SMMU. 0 = DISABLE 1 = ENABLE

18.11.1.48 MC_EMEM_ARB_MISC2_0

External Memory Arbitration Configuration: Miscellaneous Thresholds (2)

ALLOW_B2B_TA_REQS: For LPDDR4, back-to-back transactions are not allowed or masked writes will not work (i.e. ALLOW_B2B_TA_REQS must be disabled for LPDDR4).

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- The ALLOW_B2B_TA_REQS field should be saved to scratch registers and restored by the Boot ROM during warm boot.
- The other fields of this register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from other settings.

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	ALLOW_B2B_TA_REQS: [PMC] If enabled, the transaction arbiter (TA) is allowed to grant requests on back-to-back cycles. Previously, the only cases where transactions need to be issued on back-to-back cycles were as follows: EMC to MC clock ratio DRAM burst length 4:1 BL=8 2:1 BL=4 Tegra X1 devices do not support 4:1 EMC to MC clock ratio or BL=4, so there is no need to be able to issue transactions on back-to-back cycles. When transactions are only allowed to issue every other clock, both the deadline and the masked_write information should always be correct. This bit is required to be disabled for LPDDR4 since proper masked_write operation is required. 0 = DISABLE 1 = ENABLE

18.11.1.49 MC_EMEM_ARB_RING1_THROTTLE_0

External Memory Arbitration Configuration: Snap Arbiter Throttle

Ring1 Snap Arbiter Throttle - Inserts some number of stall cycles at Ring1 after every request.

Throttle cycle count varies whether outstanding_request count is below (LOW) or above (HIGH) the max_outstanding threshold.

Boot requirements: During Boot ROM section of warm boot, this register may be derived from other settings.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0xe0 | Read/Write: R/W | Reset: 0x001f0000 (0bxxxxxxxxxxxx11111xxxxxxxxxxx00000)

Bit	Reset	Description
20:16	0x1f	RING1_THROTTLE_CYCLES_HIGH: Cycles of throttle after each request when outstanding transaction count is greater than or equal to threshold.
4:0	0x0	RING1_THROTTLE_CYCLES_LOW: Cycles of throttle after each request when outstanding transaction count is below threshold. Suggested programming: (DIV==1)? 1 : 0.

18.11.1.50 MC_EMEM_ARB_RING3_THROTTLE_0

External Memory Arbitration Configuration: Snap Arbiter Throttle

Highest Ring Snap Arbiter Throttle - Although called ring3, this register works on the highest ring.

Boot requirements: This register should be saved in the SDRAM and restored by the OS during warm boot.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	RING3_THROTTLE_CYCLES: Cycles of throttle after each request.

18.11.1.51 MC_EMEM_ARB_OVERRIDE_0

External Memory Arbitration Configuration: Feature Overrides

Boot requirements:

- Some fields of this register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0xe8 | Read/Write: R/W | Reset: 0x10000000 (0b0xx100xx00000000000000000xxx00xxx)

Bit	Reset	Description
31	DISABLE	ARB_HUM_FIFO_DEADLOCK_CHECK_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
28	ENABLE	ARB_EMEM_SPO_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
27	DISABLE	ARB_EMEM_AP_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
26	DISABLE	ARB_HUM_FIFO_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
23	DISABLE	ISO_TA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
22	DISABLE	ISO_DA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
21	DISABLE	ISO_BC_INHERIT_ON_PRIINV_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
20	DISABLE	ISO_BC_CAUSE_PRIINV_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
19	DISABLE	ISO_BA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
18	DISABLE	ISO_AA_OVERRIDE 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
17	DISABLE	ARB_EMEM_BUBBLECALC_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
16	DISABLE	ALLOC_ONE_BQ_PER_CLIENT: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
15	DISABLE	PRIORITY_INVERSION_ISO_THRESHOLD_BUS_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
14	DISABLE	PRIORITY_INVERSION_ISO_THRESHOLD_BANK_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
13	DISABLE	PRIORITY_INVERSION_THRESHOLD_BUS_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE

Bit	Reset	Description
12	DISABLE	PRIORITY_INVERSION_THRESHOLD_BANK_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
11	DISABLE	PRIORITY_INVERSION_EQ_PRI_LEN_LIMIT_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
10	DISABLE	OBSERVED_DIRECTION_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
9	DISABLE	BC2AA_HOLDOFF_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
8	DISABLE	TS2AA_HOLDOFF_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
4	DISABLE	EXPIRE_UPDATE_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
3	DISABLE	GPU_SLICE_MERGE_OVERRIDE: [PMC] Overrides the ability to combine GPU traffic from multiple slices into a single bank queue. When set to OVERRIDE, each GPU slice will be considered as a different client, and thus use its own bank queue. When set to DISABLE (i.e., feature is not overridden), all GPU slices will be considered to be a "virtual client" and combined into a single bank queue 0 = DISABLE 1 = ENABLE 1 = OVERRIDE

18.11.1.52 MC_EMEM_ARB_RSV_0

EMEM Arbiter Reserved

Boot requirements:

- Some fields of this register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- This register should be saved in the SDRAM and restored by the OS during warm boot.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0xec | Read/Write: R/W | Reset: 0xff00ff00 (0b11111111000000001111111100000000)

Bit	Reset	Description
31:24	0xff	EMEM_ARB_RESERVED_BYTE3
23:16	0x0	EMEM_ARB_RESERVED_BYTE2
15:8	0xff	EMEM_ARB_RESERVED_BYTE1
7:0	0x0	EMEM_ARB_RESERVED_BYTE0

18.11.1.53 MC_CLKEN_OVERRIDE_0

Second-Level Clock Enable Overrides

Boot requirements: This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0xf4 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx10000000000x00x0)

Bit	Reset	Description
15	CLK_OVERRIDE	EMC_RESP_CLKEN_OVR: EMC Response clock: If Overridden, mc_emcresp_clk = mc_all_clk 0 = CLK_GATED 1 = CLK_OVERRIDE 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
14	CLK_GATED	TSR_CLKEN_OVR: EARB TSR clock: If Overridden, mc_tsr_clk = mc_earb_clk 0 = CLK_GATED 1 = CLK_OVERRIDE 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
13	CLK_GATED	RING2_CLKEN_OVR: Ring2 clock 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
12	CLK_GATED	RING1_CLKEN_OVR: Ring1 clock 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
11	CLK_GATED	CH_DDA_CLKEN_OVR: CIF DDA clocken 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
10	CLK_GATED	HUB_DDA_CLKEN_OVR: CIF DDA clocken 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
9	CLK_GATED	WCAM_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
8	CLK_GATED	PTC_CACHE_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
7	CLK_GATED	PTC_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
6	CLK_GATED	TLB_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
5	CLK_GATED	WB_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
3	CLK_GATED	REGS_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
2	CLK_GATED	EARB_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
0	CLK_GATED	CIF_CLKEN_OVR : 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

18.11.1.54 MC_TIMING_CONTROL_0

Shadowed Registers: Update Trigger

Boot requirements: Writing this register with 0x1 will trigger a timing update event, which should be used in both warm boot and cold boot sequences.

Offset: 0xfc | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TIMING_UPDATE

18.11.1.55 MC_STAT_CONTROL_0

Statistics: Control

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	DISABLE	EMC_PM_STOP_TRIGGER: 0 = DISABLE 1 = ENABLE
2	DISABLE	EMC_PM_START_TRIGGER: 0 = DISABLE 1 = ENABLE
1:0	RST	EMC_GATHER: 0 = RST 2 = DISABLE 3 = ENABLE

18.11.1.56 MC_CLIENT_HOTRESET_CTRL_0

Writing FLUSH_ENABLE to a bit in this register causes a flush to be performed on all of the clients for the selected swname. The status of the flush (done/in progress) can be monitored in the CLIENT_HOTRESET_STATUS register.

A proper client reset sequence is as follows:

1. Set the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to block further access by the client, and start the flush process.
2. Poll the CLIENT_HOTRESET_STATUS register until the appropriate bit returns FLUSH_DONE.
3. Clear module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to reset the module.
4. Set module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to release the module reset.
5. Clear the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to allow transactions to flow.

Memory Client Hot Reset Control Register

Offset: 0x200 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxx000000x00xx0x0000xx0000)

Bit	Reset	Description
31	DISABLE	SDMMC3A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
30	DISABLE	SDMMC2A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
29	DISABLE	SDMMC1A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
22	DISABLE	TSEC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
20	DISABLE	XUSB_DEV_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
19	DISABLE	XUSB_HOST_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
18	DISABLE	VIC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
17	DISABLE	VI_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
15	DISABLE	SATA_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
14	DISABLE	PPCS_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
11	DISABLE	NVENC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
9	DISABLE	MPCORE_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
8	DISABLE	ISP2_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
7	DISABLE	HDA_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
6	DISABLE	HC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
3	DISABLE	DCB_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
2	DISABLE	DC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
1	DISABLE	AVPC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
0	DISABLE	AFI_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

18.11.1.57 MC_CLIENT_HOTRESET_STATUS_0

Contains one bit for each swname, indicating the status of any flush that has been requested in the CLIENT_HOTRESET_CTRL register.

Note: *If no flush has been requested, this register returns FLUSH_DONE.*

Memory Client Hot Reset Status Register

Offset: 0x204 | Read/Write: RO | Reset: 0xX0XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC3A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
30	X	SDMMC2A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
29	X	SDMMC1A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
22	X	TSEC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
20	X	XUSB_DEV_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
19	X	XUSB_HOST_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
18	X	VIC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
17	X	VI_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
15	X	SATA_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
14	X	PPCS_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
11	X	NVENC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
9	X	MPCORE_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
8	X	ISP2_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
7	X	HDA_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
6	X	HC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
3	X	DCB_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
2	X	DC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
1	X	AVPC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
0	X	AFI_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS

18.11.1.58 MC_EMEM_ARB_ISOCHRONOUS_0_0

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type. This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Isochronous Settings

Offset: 0x208 | Read/Write: R/W | Reset: 0x0023007e (0b0000xxxx001xxx1100xxxxxxx1111110

Bit	Reset	Description
31	DISABLE	ISO_SATAR: client satar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
30	DISABLE	ISO_PPCSAHBSLVR: client ppcsahbslvr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
29	DISABLE	ISO_PPCSAHBDMAR: client ppcsahbdmar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
28	DISABLE	ISO_NVENCSRD: client nvencsrdis treated as an isochronous client 0 = DISABLE 1 = ENABLE
23	DISABLE	ISO_HOST1XR: client host1xr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
22	DISABLE	ISO_HOST1XDMAR: client host1xdmar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
21	DISABLE	ISO_HDAR: client hdar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
17	0x1	ISO_DISPLAYHCB: client displayhcb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
16	0x1	ISO_DISPLAYHC: client displayhc is treated as an isochronous client 0 = DISABLE 1 = ENABLE
15	DISABLE	ISO_AVPCARM7R: client avpcarm7r is treated as an isochronous client 0 = DISABLE 1 = ENABLE
14	DISABLE	ISO_AFIR: client afir is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	0x1	ISO_DISPLAY0CB: client display0cb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
5	0x1	ISO_DISPLAY0C: client display0c is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	0x1	ISO_DISPLAY0BB: client display0bb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
3	0x1	ISO_DISPLAY0B: client display0b is treated as an isochronous client 0 = DISABLE 1 = ENABLE
2	0x1	ISO_DISPLAY0AB: client display0ab is treated as an isochronous client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x1	ISO_DISPLAY0A: client display0a is treated as an isochronous client 0 = DISABLE 1 = ENABLE
0	0x1	ISO_PTCR: client ptcr is treated as an isochronous client 0 = DISABLE 1 = ENABLE

18.11.1.59 MC_EMEM_ARB_ISOCHRONOUS_1_0

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type.

This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client isochronous settings

Offset: 0x20c | Read/Write: R/W | Reset: 0x00000000 (0bxx000x0xx00xx00xxxxx0xxx0xxxxxxx)

Bit	Reset	Description
29	DISABLE	ISO_SATAW: client sataw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
28	DISABLE	ISO_PPCSAHBSLVW: client ppcsahbslvw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
27	DISABLE	ISO_PPCSAHBDMAW: client ppcsahbdmaw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
25	DISABLE	ISO_MPCOREW: client mpcorew is treated as an isochronous client 0 = DISABLE 1 = ENABLE
22	DISABLE	ISO_HOST1XW: client host1xw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
21	DISABLE	ISO_HDAW: client hdaw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
18	DISABLE	ISO_AVPCARM7W: client avpcarm7w is treated as an isochronous client 0 = DISABLE 1 = ENABLE
17	DISABLE	ISO_AFIW: client afiw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
11	DISABLE	ISO_NVENCSWR: client nvencswris treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	DISABLE	ISO_MPCORER: client mpcorer is treated as an isochronous client 0 = DISABLE 1 = ENABLE

18.11.1.60 MC_EMEM_ARB_ISOCHRONOUS_2_0

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type.

This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Isochronous Settings

Offset: 0x210 | Read/Write: R/W | Reset: 0x04000000 (0bxxxxx1000000xx00x00000xx00x0xxxx)

Bit	Reset	Description
26	0x1	ISO_DISPLAYT: Client displayt is treated as an isochronous client 0 = DISABLE 1 = ENABLE
25	DISABLE	ISO_GPUSWR: Client gpuswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
24	DISABLE	ISO_GPUSRD: Client gpusrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
21	DISABLE	ISO_TSECSWR: Client tsecswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
20	DISABLE	ISO_TSECSRD: Client tsecsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
17	DISABLE	ISO_ISPWBB: Client emucifwispwbb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
16	DISABLE	ISO_ISPWAB: Client ispwab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
14	DISABLE	ISO_ISPRAB: Client isprab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
13	DISABLE	ISO_XUSB_DEVW: Client xusb_devw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
12	DISABLE	ISO_XUSB_DEVR: Client xusb_devr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
11	DISABLE	ISO_XUSB_HOSTW: Client xusb_hostw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
10	DISABLE	ISO_XUSB_HOSTR: Client xusb_hostr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	DISABLE	ISO_ISPWB: Client ispwb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	DISABLE	ISO_ISPWA: Client ispwa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	DISABLE	ISO_ISPRA: Client ispra is treated as an isochronous client 0 = DISABLE 1 = ENABLE

18.11.1.61 MC_EMEM_ARB_ISOCHRONOUS_3_0

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type. This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Isochronous Settings

Offset: 0x214 | Read/Write: R/W | Reset: 0x00080000 (0b00xx0000xxxx10xxxx00xxxx00000000)

Bit	Reset	Description
31	DISABLE	ISO_NVJPGSWR: client nvjpgswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
30	DISABLE	ISO_NVJPGSRD: client nvjpgsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
27	DISABLE	ISO_APEW: client apew is treated as an isochronous client 0 = DISABLE 1 = ENABLE
26	DISABLE	ISO_APER: client aper is treated as an isochronous client 0 = DISABLE 1 = ENABLE
25	DISABLE	ISO_NVDECSWR: client nvdecswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
24	DISABLE	ISO_NVDECSRD: client nvdecsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
19	0x1	ISO_DISPLAYD: Client displayd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
18	DISABLE	ISO_VIW: Client viw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
13	DISABLE	ISO_VICSWR: Client vicswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
12	DISABLE	ISO_VICSRD: Client vicsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	DISABLE	ISO_SDMMCWAB: Client sdmmcwab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	DISABLE	ISO_SDMMCW: Client sdmmcw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
5	DISABLE	ISO_SDMMCWAA: Client sdmmcwaa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	DISABLE	ISO_SDMMCWA: Client sdmmcwa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
3	DISABLE	ISO_SDMMCRAB: Client sdmmcrab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
2	DISABLE	ISO_SDMMCR: Client sdmmcr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
1	DISABLE	ISO_SDMMCRAA: Client sdmmcraa is treated as an isochronous client 0 = DISABLE 1 = ENABLE
0	DISABLE	ISO_SDMMCRA: Client sdmmcra is treated as an isochronous client 0 = DISABLE 1 = ENABLE

18.11.1.62 MC_EMEM_ARB_HYSTERESIS_0_0

Controls whether a client is considered to be a hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Hysteresis Settings

Offset: 0x218 | Read/Write: R/W | Reset: 0x0003007e (0b0000xxxx000xxx1100xxxxxxx1111110)

Bit	Reset	Description
31	DISABLE	HYST_SATAR: client satar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
30	DISABLE	HYST_PPCSAHBSLVR: client ppcsahbslvr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
29	DISABLE	HYST_PPCSAHBDMAR: client ppcsahbdmar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
28	DISABLE	HYST_NVENCSRD: client nvencsrdis treated as a hysteresis client 0 = DISABLE 1 = ENABLE
23	DISABLE	HYST_HOST1XR: client host1xr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
22	DISABLE	HYST_HOST1XDMAR: client host1xdmar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
21	DISABLE	HYST_HDAR: client hdar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
17	0x1	HYST_DISPLAYHCB: client displayhcb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
16	0x1	HYST_DISPLAYHC: client displayhc is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
15	DISABLE	HYST_AVPCARM7R: client avpcarm7r is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
14	DISABLE	HYST_AFIR: client afir is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
6	0x1	HYST_DISPLAY0CB: client display0cb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
5	0x1	HYST_DISPLAY0C: client display0c is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	0x1	HYST_DISPLAY0BB: client display0bb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	0x1	HYST_DISPLAY0B: client display0b is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	0x1	HYST_DISPLAY0AB: client display0ab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x1	HYST_DISPLAY0A: client display0a is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
0	DISABLE	HYST_PTCR: client ptcr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.63 MC_EMEM_ARB_HYSTERESIS_1_0

Controls whether a client is considered to be an hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client Hysteresis Settings

Offset: 0x21c | Read/Write: R/W | Reset: 0x00000000 (0bxx000x0xx00xx00xxxxx0xxx0xxxxxxx)

Bit	Reset	Description
29	DISABLE	HYST_SATAW: client sataw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
28	DISABLE	HYST_PPCSAHBSLVW: client ppcsahbslvw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
27	DISABLE	HYST_PPCSAHBDMAW: client ppcsahbdmaw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
25	DISABLE	HYST_MPCOREW: client mpcorew is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
22	DISABLE	HYST_HOST1XW: client host1xw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
21	DISABLE	HYST_HDAW: client hdaw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
18	DISABLE	HYST_AVPCARM7W: client avpcarm7w is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
17	DISABLE	HYST_AFIW: client afiw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
11	DISABLE	HYST_ NVENCSWR: client nvencswris treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	DISABLE	HYST_MPCORER: client mpcorer is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.64 MC_EMEM_ARB_HYSTERESIS_2_0

Controls whether a client is considered to be a hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client hysteresis settings

Offset: 0x220 | Read/Write: R/W | Reset: 0x04000000 (0bxxxxx1000000xx00x00000xx00x0xxxx)

Bit	Reset	Description
26	0x1	HYST_DISPLAYT: client displayt is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
25	DISABLE	HYST_GPUSWR: client gpuswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
24	DISABLE	HYST_GPUSRD: client gpusrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
21	DISABLE	HYST_TSECSWR: client tsecswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
20	DISABLE	HYST_TSECSRD: client tsecsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
17	DISABLE	HYST_ISPWBB: client ispwbb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
16	DISABLE	HYST_ISPWAB: client ispwab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
14	DISABLE	HYST_ISPRAB: client isprab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
13	DISABLE	HYST_XUSB_DEVW: client xusb_devw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
12	DISABLE	HYST_XUSB_DEVR: client xusb_devr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
11	DISABLE	HYST_XUSB_HOSTW: client xusb_hostw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
10	DISABLE	HYST_XUSB_HOSTR: client xusb_hostr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	DISABLE	HYST_ISPWB: client ispwb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
6	DISABLE	HYST_ISPWA: client ispwa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	DISABLE	HYST_ISPRA: client ispra is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.65 MC_EMEM_ARB_HYSTERESIS_3_0

Controls whether a client is considered to be an hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client hysteresis settings

Offset: 0x224 | Read/Write: R/W | Reset: 0x00080000 (0b00xx0000xxxx10xxxx00xxxx00000000)

Bit	Reset	Description
31	DISABLE	HYST_NVJPGSWR: client nvjpgswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
30	DISABLE	HYST_NVJPGSRD: client nvjpgsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
27	DISABLE	HYST_APEW: client apew is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
26	DISABLE	HYST_APER: client aper is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
25	DISABLE	HYST_NVDECSWR: client nvdecswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
24	DISABLE	HYST_NVDECSRD: client nvdecsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
19	0x1	HYST_DISPLAYD: client displayd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
18	DISABLE	HYST_VIW: client viw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
13	DISABLE	HYST_VICSWR: client vicswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
12	DISABLE	HYST_VICSRD: client vicsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	DISABLE	HYST_SDMMCWAB: client sdmmcwab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
6	DISABLE	HYST_SDMMCW: client sdmmcw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
5	DISABLE	HYST_SDMMCWAA: client sdmmcwaa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	DISABLE	HYST_SDMMCWA: client sdmmcwa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	DISABLE	HYST_SDMMCRAB: client sdmmcrab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	DISABLE	HYST_SDMMCR: client sdmmcr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
1	DISABLE	HYST_SDMMCRAA: client sdmmcraa is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
0	DISABLE	HYST_SDMMCRA: client sdmmcra is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.66 MC_SMMU_TRANSLATION_ENABLE_0_0

Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client SMMU translation enables

Offset: 0x228 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xf0e3c07e (0b1111xxxx111xxx1111xxxxxxx111111x)

Bit	Reset	Description
31	ENABLE	SMMU_SATAR_ENABLE: enable client satar to be translated by SMMU 0 = DISABLE 1 = ENABLE
30	ENABLE	SMMU_PPCSAHBSLVR_ENABLE: enable client ppcsahbslvr to be translated by SMMU 0 = DISABLE 1 = ENABLE
29	ENABLE	SMMU_PPCSAHBDMAR_ENABLE: enable client ppcsahbdmar to be translated by SMMU 0 = DISABLE 1 = ENABLE
28	ENABLE	SMMU_NVENCSRD_ENABLE: enable client nvencsrdto be translated by SMMU 0 = DISABLE 1 = ENABLE
23	ENABLE	SMMU_HOST1XR_ENABLE: enable client host1xr to be translated by SMMU 0 = DISABLE 1 = ENABLE
22	ENABLE	SMMU_HOST1XDMAR_ENABLE: enable client host1xdmar to be translated by SMMU 0 = DISABLE 1 = ENABLE
21	ENABLE	SMMU_HDAR_ENABLE: enable client hdar to be translated by SMMU 0 = DISABLE 1 = ENABLE
17	ENABLE	SMMU_DISPLAYHCB_ENABLE: enable client displayhcb to be translated by SMMU 0 = DISABLE 1 = ENABLE
16	ENABLE	SMMU_DISPLAYHC_ENABLE: enable client displayhc to be translated by SMMU 0 = DISABLE 1 = ENABLE
15	ENABLE	SMMU_AVPCARM7R_ENABLE: enable client avpcarm7r to be translated by SMMU 0 = DISABLE 1 = ENABLE
14	ENABLE	SMMU_AFIR_ENABLE: enable client afir to be translated by SMMU 0 = DISABLE 1 = ENABLE
6	ENABLE	SMMU_DISPLAY0CB_ENABLE: enable client display0cb to be translated by SMMU 0 = DISABLE 1 = ENABLE
5	ENABLE	SMMU_DISPLAY0C_ENABLE: enable client display0c to be translated by SMMU 0 = DISABLE 1 = ENABLE
4	ENABLE	SMMU_DISPLAY0BB_ENABLE: enable client display0bb to be translated by SMMU 0 = DISABLE 1 = ENABLE
3	ENABLE	SMMU_DISPLAY0B_ENABLE: enable client display0b to be translated by SMMU 0 = DISABLE 1 = ENABLE
2	ENABLE	SMMU_DISPLAY0AB_ENABLE: enable client display0ab to be translated by SMMU 0 = DISABLE 1 = ENABLE
1	ENABLE	SMMU_DISPLAY0A_ENABLE: enable client display0a to be translated by SMMU 0 = DISABLE 1 = ENABLE

18.11.1.67 MC_SMMU_TRANSLATION_ENABLE_1_0

Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client SMMU translation enables

Offset: 0x22c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x38660800 (0bxx111xxxx11xx11xxxxx1xxxxxxxxxxx)

Bit	Reset	Description
29	ENABLE	SMMU_SATAW_ENABLE: enable client sataw to be translated by SMMU 0 = DISABLE 1 = ENABLE
28	ENABLE	SMMU_PPCSAHBSLVW_ENABLE: enable client ppcsahbslvw to be translated by SMMU 0 = DISABLE 1 = ENABLE
27	ENABLE	SMMU_PPCSAHBDMAW_ENABLE: enable client ppcsahbdmaw to be translated by SMMU 0 = DISABLE 1 = ENABLE
22	ENABLE	SMMU_HOST1XW_ENABLE: enable client host1xw to be translated by SMMU 0 = DISABLE 1 = ENABLE
21	ENABLE	SMMU_HDAW_ENABLE: enable client hdaw to be translated by SMMU 0 = DISABLE 1 = ENABLE
18	ENABLE	SMMU_AVPCARM7W_ENABLE: enable client avpcarm7w to be translated by SMMU 0 = DISABLE 1 = ENABLE
17	ENABLE	SMMU_AFIW_ENABLE: enable client afiw to be translated by SMMU 0 = DISABLE 1 = ENABLE
11	ENABLE	SMMU_NVENCSWR_ENABLE: enable client nvencswrto be translated by SMMU 0 = DISABLE 1 = ENABLE

18.11.1.68 MC_SMMU_TRANSLATION_ENABLE_2_0

Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-client SMMU translation enables

Offset: 0x230 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0Xf37cd0 (0bxxxxx1xx1111xx11x11111xx11x1xxxx)

Bit	R/W	Reset	Description
26	RW	ENABLE	SMMU_DISPLAYT_ENABLE: enable client displayt to be translated by the SMMU 0 = DISABLE 1 = ENABLE
25	RO	X	SMMU_GPUSWR_ENABLE: GPU translation is handled by swid, so this is required to be enabled 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
24	RO	X	SMMU_GPUSRD_ENABLE: GPU translation is handled by swid, so this is required to be enabled 0 = DISABLE 1 = ENABLE
21	RW	ENABLE	SMMU_TSECSWR_ENABLE: enable client tsecswr to be translated by the SMMU 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	MMU_TSECSRD_ENABLE: enable client tsecsrd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
17	RW	ENABLE	SMMU_ISPWBB_ENABLE: enable client ispwbb to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
16	RW	ENABLE	SMMU_ISPWAB_ENABLE: enable client ispwab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SMMU_ISPRAB_ENABLE: enable client isprab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
13	RW	ENABLE	SMMU_XUSB_DEVW_ENABLE: enable client xusb_devw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SMMU_XUSB_DEVR_ENABLE: enable client xusb_devr to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
11	RW	ENABLE	SMMU_XUSB_HOSTW_ENABLE: enable client xusb_hostw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
10	RW	ENABLE	SMMU_XUSB_HOSTR_ENABLE: enable client xusb_hostr to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SMMU_ISPWB_ENABLE: enable client ispwb to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SMMU_ISPWA_ENABLE: enable client ispwa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
4	RW	ENABLE	SMMU_ISPRA_ENABLE: enable client ispra to be translated by the SMMU. 0 = DISABLE 1 = ENABLE

18.11.1.69 MC_SMMU_TRANSLATION_ENABLE_3_0

Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Per-Client SMMU Translation Enables

Offset: 0x234 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xcf0c30ff (0b11xx1111xxxx11xxxx11xxxx11111111)

Bit	Reset	Description
31	ENABLE	SMMU_NVJPGSWR_ENABLE: enable client nvjpgswr to be translated by SMMU 0 = DISABLE 1 = ENABLE
30	ENABLE	SMMU_NVJPGSRD_ENABLE: enable client nvjpgsrd to be translated by SMMU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	ENABLE	SMMU_APEW_ENABLE: enable client apew to be translated by SMMU 0 = DISABLE 1 = ENABLE
26	ENABLE	SMMU_APER_ENABLE: enable client aper to be translated by SMMU 0 = DISABLE 1 = ENABLE
25	ENABLE	SMMU_NVDECSWR_ENABLE: enable client nvdecswr to be translated by SMMU 0 = DISABLE 1 = ENABLE
24	ENABLE	SMMU_NVDECSRD_ENABLE: enable client nvdecsrd to be translated by SMMU 0 = DISABLE 1 = ENABLE
19	ENABLE	SMMU_DISPLAYD_ENABLE: enable client displayd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
18	ENABLE	SMMU_VIW_ENABLE: enable client viw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
13	ENABLE	SMMU_VICSRD_ENABLE: enable client vicsrd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
12	ENABLE	SMMU_VICSRD_ENABLE: enable client vicsrd to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
7	ENABLE	SMMU_SDMMCWAB_ENABLE: enable client sdmmcwab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
6	ENABLE	SMMU_SDMMCW_ENABLE: enable client sdmmcw to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
5	ENABLE	SMMU_SDMMCWAA_ENABLE: enable client sdmmcwaa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
4	ENABLE	SMMU_SDMMCWA_ENABLE: enable client sdmmcwa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
3	ENABLE	SMMU_SDMMCRAB_ENABLE: enable client sdmmcrab to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
2	ENABLE	SMMU_SDMMCR_ENABLE: enable client sdmmcr to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
1	ENABLE	SMMU_SDMMCRAA_ENABLE: enable client sdmmcraa to be translated by the SMMU. 0 = DISABLE 1 = ENABLE
0	0x0	SMMU_SDMMCRA_ENABLE: enable client sdmmcra to be translated by the SMMU. 0 = DISABLE 1 = ENABLE

18.11.1.70 MC_SMMU_AFI_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU AFI ASID Assignment Register

Offset: 0x238 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	AFI_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	AFI_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	AFI_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	AFI_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	AFI_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.71 MC_SMMU_AVPC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU AVPC ASID Assignment Register

Offset: 0x23c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	AVPC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	AVPC_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.72 MC_SMMU_DC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU DC ASID Assignment Register

Offset: 0x240 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	DC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	DC_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	DC_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	DC_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	DC_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.73 MC_SMMU_DCB_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU DCB ASID Assignment Register

Offset: 0x244 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	DCB_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	DCB_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	DCB_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	DCB_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	DCB_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.74 MC_SMMU_HC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU HC ASID Assignment Register

Offset: 0x250 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	HC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	HC_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.75 MC_SMMU_HDA_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU HDA ASID Assignment Register

Offset: 0x254 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	HDA_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
30:24	0x0	HDA_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	HDA_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	HDA_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1

Bit	Reset	Description
6:0	0x0	HDA_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.76 MC_SMMU_ISP2_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU ISP2 ASID Assignment Register

Offset: 0x258 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	ISP2_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	ISP2_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	ISP2_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	ISP2_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	ISP2_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.77 MC_SMMU_NVENC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU MSENC NVENC Assignment Register

Offset: 0x264 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	NVENC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	NVENC_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.78 MC_SMMU_NV_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU NV ASID Assignment Register

Offset: 0x268 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	NV_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1:0	0x0	NV_ASID: ASID to use for translation, if enabled

18.11.1.79 MC_SMMU_NV2_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU NV2 ASID Assignment Register

Offset: 0x26c | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	NV2_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	NV2_ASID: ASID to use for translation, if enabled

18.11.1.80 MC_SMMU_PPCS_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU PPCS ASID Assignment Register

Offset: 0x270 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	PPCS_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	PPCS_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.81 MC_SMMU_SATA_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU SATA ASID Assignment Register

Offset: 0x274 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	SATA_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	SATA_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	SATA_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	SATA_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1

Bit	Reset	Description
6:0	0x0	SATA_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.82 MC_SMMU_VI_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU VI ASID Assignment Register

Offset: 0x280 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	VI_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	VI_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	VI_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	VI_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	VI_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.83 MC_SMMU_VIC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU VIC ASID Assignment Register

Offset: 0x284 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	VIC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	VIC_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.84 MC_SMMU_XUSB_HOST_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU XUSB_HOST ASID Assignment Register

Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	XUSB_HOST_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	XUSB_HOST_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	XUSB_HOST_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	XUSB_HOST_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	XUSB_HOST_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.85 MC_SMMU_XUSB_DEV_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU XUSB_DEV ASID Assignment Register

Offset: 0x28c | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	XUSB_DEV_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	XUSB_DEV_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	XUSB_DEV_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	XUSB_DEV_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	XUSB_DEV_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.86 MC_SMMU_TSEC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU TSEC ASID Assignment Register

Offset: 0x294 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	TSEC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	TSEC_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	TSEC_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	TSEC_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	TSEC_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.87 MC_SMMU_PPCS1_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

SMMU PPCS1ASID Assignment Register

Offset: 0x298 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	PPCS1_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	PPCS1_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.88 MC_VIDEO_PROTECT_VPR_OVERRIDE_0

MC VPR OVERRIDE Register

Offset: 0x418 | Read/Write: R/W | Reset: 0xe4bac343 (0b111xx1xx1011101x11xx0x1101xx0011)

Bit	Reset	Description
31	ENABLE	SDMMC3A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
30	ENABLE	SDMMC2A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
29	ENABLE	SDMMC1A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
26	ENABLE	DC1_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
23	ENABLE	PPCS1_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
22	DISABLE	TSEC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
20	ENABLE	XUSB_DEV_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
19	ENABLE	XUSB_HOST_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
18	DISABLE	VIC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
17	ENABLE	VI_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
15	ENABLE	SATA_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
14	ENABLE	PPCS_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	NVENC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
9	ENABLE	MPCORE_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
8	ENABLE	ISP2_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
7	DISABLE	HDA_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
6	ENABLE	HC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
3	DISABLE	DCB_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
2	DISABLE	DC_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
1	ENABLE	AVPC_VPR_OVERRIDE 0 = DISABLE 1 = ENABLE
0	ENABLE	AFI_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE

18.11.1.89 MC_VIDEO_PROTECT_VPR_OVERRIDE1_0

MC VPR OVERRIDE Register

Offset: 0x590 | Read/Write: R/W | Reset: 0x00001ed3 (0bxxxxxxxxxxxxxxxx00001111011010011)

Bit	Reset	Description
16	DISABLE	NVDEC1_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
15	DISABLE	TSECB1_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
14	DISABLE	TSEC1_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
13	DISABLE	TSECB_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
12	ENABLE	ETR_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
11	ENABLE	AXIAP_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
10	ENABLE	SE1_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
9	ENABLE	HC1_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
8	DISABLE	NVJPG_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	ENABLE	SE_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
6	ENABLE	APE_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
5	DISABLE	NVDEC_VPR_OVERRIDE: [PMC_SECURE] if set, vpr attribute is overridden for this module to 0 0 = DISABLE 1 = ENABLE
4	ENABLE	PPCS2_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
3	DISABLE	GPUB_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
2	DISABLE	GPU_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
1	ENABLE	ISP2B_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE
0	ENABLE	SDMMC4A_VPR_OVERRIDE: 0 = DISABLE 1 = ENABLE

18.11.1.90 MC_SMMU_TLB_SET_SELECTION_MASK_0_0

TLB Set Selection Mask (Bit 0)

Mask is ANDed with the Virtual Address, and the resulting value is XORed to a single bit. That bit selects either set0 or set1.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x600 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxx000000001xxxxxxxxxxxxxxx)

Bit	Reset	Description
23:15	0x1	TLB_SET_SELECTION_MASK_0

18.11.1.91 MC_DISPLAY_SNAP_RING_0

Display Arbiter Ring Selection

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0x608 | Read/Write: R/W | Reset: 0x00000003 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
31	ENABLED	DISPLAY_SNAP_RING_WRITE_ACCESS : 0 = ENABLED 1 = DISABLED
1	RING1	DISB_SNAP_RING : 0 = RING0 1 = RING1
0	RING1	DIS_SNAP_RING : 0 = RING0 1 = RING1

18.11.1.92 MC_ERR_VPR_STATUS_0

Offset: 0x654 | Read/Write: RO | Reset: 0x00XXX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21:20	X	ERR_VPR_ADR_HI: Higher address bits of erring address whose lower bits are available in the ERR_VPR_ADR register
18	X	ERR_VPR_SWAP
17	X	ERR_VPR_SECURITY: Set if transaction was secure. 0 = NONSECURE 1 = SECURE
16	X	ERR_VPR_RW: Direction of the access that caused the error. 0 = READ 1 = WRITE
14:12	X	ERR_VPR_ADR1: Second subpartition unique address bits
7:0	X	ERR_VPR_ID: Client identifier (see above list) of the access that caused the error.

18.11.1.93 MC_ERR_VPR_ADR_0

Offset: 0x658 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ERR_VPR_ADR

18.11.1.94 MC_IRAM_REG_CTRL_0

Access control register for IRAM aperture programming. This bit is "sticky." Registers are writeable by default.

Once write access is disabled, it cannot be re-enabled without a system reset

The following registers are controlled by this field:

- IRAM_BOM
- IRAM_TOM

Offset: 0x964 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	ENABLED	IRAM_CFG_WRITE_ACCESS: 0 = ENABLED 1 = DISABLED

18.11.1.95 MC_EMEM_CFG_ACCESS_CTRL_0

Access Control Bit for EMEM_CFG Registers

This bit is "sticky." Registers are writeable by default. Once write access is disabled, it cannot be re-enabled without a system reset.

The following registers are controlled by this register:

- EMEM_CFG
- EMEM_ADR_CFG
- EMEM_ADR_CFG_DEV0
- EMEM_ADR_CFG_DEV1
- EMEM_ADR_CFG_CHANNEL_MASK
- EMEM_ADR_CFG_BANK_MASK_0
- EMEM_ADR_CFG_BANK_MASK_1

- EMEM_ADR_CFG_BANK_MASK_2
- EMEM_BANK_SWIZZLE_CFG0
- EMEM_BANK_SWIZZLE_CFG1
- EMEM_BANK_SWIZZLE_CFG2
- EMEM_BANK_SWIZZLE_CFG3

Offset: 0x664 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	ENABLED	EMEM_CFG_WRITE_ACCESS: 0 = ENABLED 1 = DISABLED

18.11.1.96 MC_TZ_SECURITY_CTRL_0

Trustzone Security Control Register

Controls "Strict"ness of TrustZone security check.

This register is "sticky" - once the strict check is enabled, it cannot be disabled without a reset.

Offset: 0x668 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLED	CPU_STRICT_TZ_APERTURE_CHECK: When this bit is set to ENABLED, the TrustZone security check requires that the CPU's security state bit exactly match the aperture. That is, when the CPU sets the NS bit to 0, it can ONLY access the TrustZone secured aperture. When the NS bit is 1, the CPU may NOT access the TrustZone secured aperture. Any violation causes a security violation to be flagged. When this bit is set to DISABLED, the security check is relaxed. In this mode, the CPU can access memory outside of the TrustZone secured aperture even though the NS bit is set to 0 (that is, even though the CPU is in secure mode). The above description only applies to the CPU complex clients (read and write). Other clients operate in the relaxed mode. 0 = DISABLED 1 = ENABLED

18.11.1.97 MC_EMEM_ARB_OUTSTANDING_REQ_RING3_0

External Memory Arbitration Configuration: Highest Ring Outstanding Request Limiter

> ***Note:*** *Although called ring3, this register works on the highest ring.*

This register is used to limit the number of outstanding requests in the arbiter and monitor the count. If the outstanding transactions are greater than the maximum, the highest ring requests are throttled based on the MC_EMEM_ARB_RING3_THROTTLE_0 register.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During the Boot ROM section of warm boot, this register can be derived from other MC settings.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers.*

Offset: 0x66c | Read/Write: R/W | Reset: 0x8XXX0080 (0b10xxxxxxxxxxxxxxxxxxxxx010000000)

Bit	R/W	Reset	Description
31	RW	ENABLE	LIMIT_OUTSTANDING_RING3: When ENABLED, requests into ring3 are throttled after the request count reaches ARB_MAX_OUTSTANDING_RING3. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	LIMIT_DURING_HOLDOFF_OVERRIDE_RING3: When DISABLED, overrides the limiting of ring3 transactions during holdoff to let requests flow freely. 0 = DISABLE 1 = ENABLE
24:16	RO	X	ARB_OUTSTANDING_COUNT_RING3: Current number of outstanding requests
8:0	RW	0x80	ARB_MAX_OUTSTANDING_RING3: The maximum number of requests allowed in the arbiter when LIMIT_OUTSTANDING is set to ENABLE

18.11.1.98 MC_SEC_CARVEOUT_BOM_0

Offset: 0x670 | Read/Write: R/W | Reset: 0xfff00000 (0b111111111111xxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:20	0xfff	SEC_CARVEOUT_BOM: [PMC_SECURE] Base address for the SEC carveout address space.

18.11.1.99 MC_SEC_CARVEOUT_SIZE_MB_0

Offset: 0x674 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	SEC_CARVEOUT_SIZE_MB: [PMC_SECURE] SEC_CARVEOUT_SIZE_MB is the size, in megabytes, of the SEC carveout region. If set to 0, the security check in MC is disabled.

18.11.1.100 MC_SEC_CARVEOUT_REG_CTRL_0

Offset: 0x678 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	ENABLED	SEC_CARVEOUT_WRITE_ACCESS: [PMC_SECURE] Sticky bit to control the writes to the other Sec Carveout aperture registers. 0 = ENABLED 1 = DISABLED

18.11.1.101 MC_ERR_SEC_STATUS_0

Offset: 0x67c | Read/Write: RO | Reset: 0x00XXX0XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21:20	X	ERR_SEC_ADR_HI: Higher address bits of erring address whose lower bits are available in the ERR_ADR register
18	X	ERR_SEC_SWAP
17	X	ERR_SEC_SECURITY: Set if the transaction was secure. 0 = NONSECURE 1 = SECURE
16	X	ERR_SEC_RW: Direction of the access that caused the error. 0 = READ 1 = WRITE
14:12	X	ERR_SEC_ADR1: Second subpartition unique address bits.
6:0	X	ERR_SEC_ID: Client identifier (see above list) of the access that caused the error.

18.11.1.102 MC_ERR_SEC_ADR_0

Offset: 0x680 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ERR_SEC_ADR

18.11.1.103 MC_PC_IDLE_CLOCK_GATE_CONFIG_0

Partition Idle Clock Gate Config

Boot requirements:

This register should be saved in the SDRAM and restored by the OS during warm boot.

Offset: 0x684 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	CLOCK_GATE_IDLE_TICKS: Number of idle "ticks" before a partition client is clock-gated

18.11.1.104 MC_STUTTER_CONTROL_0

Offset: 0x688 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SLOW_EMCCLK_DURING_DSR: When ENABLED, the clock to the EMC will be slowed during dynamic self refresh. 0 = DISABLED 1 = ENABLED

18.11.1.105 MC_EMEM_ARB_NISO_THROTTLE_0

External Memory Arbitration Configuration: Snap Arbiter Throttle

Highest Ring Snap Arbiter Input Throttle - This register applies to the inputs of the highest ring. Refer to the EMEM_ARB_NISO_THROTTLE_MASK register to see how partition clients are configured into the "NISO meta-client" group for the purpose of NISO client throttling.

Boot requirements:

- This register should be saved in the SDRAM and restored by the OS during warm boot.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x6b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxx00000)

Bit	Reset	Description
20:16	0x0	NISO_THROTTLE_CYCLES_HIGH: Cycles of throttle after each request when the outstanding transaction count is greater than or equal to the threshold.
4:0	0x0	NISO_THROTTLE_CYCLES: Cycles of throttle after each request.

18.11.1.106 MC_EMEM_ARB_OUTSTANDING_REQ_NISO_0

External Memory Arbitration Configuration: Highest Ring Input NISO Outstanding Request Limiter

Although called NISO, this register applies to the inputs of the highest snap-arbiter ring. Refer to the EMEM_ARB_NISO_THROTTLE_MASK register to see how partition clients are configured into the "NISO meta-client" group for the purpose of NISO client throttling.

This register can be used to limit the number of outstanding requests in the arbiter and monitor the count. If the outstanding transactions are greater than the maximum, the highest ring requests are throttled based on the MC_EMEM_ARB_NISO_THROTTLE register.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During the Boot ROM section of warm boot, this register may be derived from other MC settings.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x6b4 | Read/Write: R/W | Reset: 0x8XXX0080 (0b10xxxxxxxxxxxxxxxxxxxx010000000)

Bit	R/W	Reset	Description
31	RW	ENABLE	LIMIT_OUTSTANDING_NISO: When ENABLED, requests into NISO are throttled after the request count reaches ARB_MAX_OUTSTANDING_NISO. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	LIMIT_DURING_HOLDOFF_OVERRIDE_NISO: When DISABLED, overrides the limiting of NISO transactions during holdoff to let requests flow freely. 0 = DISABLE 1 = ENABLE
24:16	RO	X	ARB_OUTSTANDING_COUNT_NISO: Current number of outstanding requests.
8:0	RW	0x80	ARB_MAX_OUTSTANDING_NISO: The maximum number of requests allowed in the arbiter when LIMIT_OUTSTANDING is set to ENABLE.

18.11.1.107 MC_EMEM_ARB_NISO_THROTTLE_MASK_0

External Memory Arbitration Configuration: Highest Ring Input NISO Throttle Mask

This register is used to group partition clients in the highest level snap-arbiter ring into a "NISO meta-client" group for the purpose of throttling. This allows soft-ISO clients in this ring to meet their bandwidth needs.

To prevent non-ISO (NISO) clients from filling the MC row-sorter (which could prevent soft-ISO clients from getting bandwidth), selective throttling is implemented at the input of ring2. To do this, the EMEM_ARB_NISO_THROTTLE_MASK register is used to specify which partition clients get included into a "NISO meta-client" group. To implement throttling, a total number of outstanding transactions counter (for example, only one counter exists in the whole "NV_MC_cif" to track this, and is shared by all throttling circuits) gets compared to a programmable maximum limit for this NISO meta-client group. If the number is greater than the maximum limit, all clients included in the "NISO meta-client" group get throttled based on the EMEM_ARB_NISO_THROTTLE register. This NISO_THROTTLE register is used to specify a number of stall cycles that get inserted at the input to the ring after every request from any of the clients included in the NISO group.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0x6b8 | Read/Write: R/W | Reset: 0x00000000 (0bxx000xxx00xx0x0xxxxxxx0x0xxx000x)

Bit	Reset	Description
29	DISABLED	NISO_THROTTLE_MASK_VICPC: If enabled, include VICPC in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
28	DISABLED	NISO_THROTTLE_MASK_USBD: If enabled, include USBD in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
27	DISABLED	NISO_THROTTLE_MASK_HOST: If enabled, include HOST in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
23	DISABLED	NISO_THROTTLE_MASK_SD: If enabled, include SD in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
22	DISABLED	NISO_THROTTLE_MASK_GK: If enabled, include GK in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
19	DISABLED	NISO_THROTTLE_MASK_MSE: If enabled, include MSE in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
17	DISABLED	NISO_THROTTLE_MASK_USBX: If enabled, include USBX in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
9	DISABLED	NISO_THROTTLE_MASK_SAX: If enabled, include SAX in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
7	DISABLED	NISO_THROTTLE_MASK_PCX: If enabled, include PCX in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
3	DISABLED	NISO_THROTTLE_MASK_AVP: If enabled, include BPMP-Lite in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
2	DISABLED	NISO_THROTTLE_MASK_APB: If enabled, include APB in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED
1	DISABLED	NISO_THROTTLE_MASK_AHB: If enabled, include AHB in the NISO meta-client group for throttling. 0 = DISABLED 1 = ENABLED

18.11.1.108 MC_EMEM_ARB_RING0_THROTTLE_MASK_0

External Memory Arbitration Configuration: Ring0 Input Throttle Mask

This register is used to group partition clients at the input to snaparb ring0 into a single "meta-client" group for throttling. This allows clients in this ring to meet their bandwidth needs.

Boot requirements:

- This register should be saved to SDRAM registers and restored by the OS during warm boot.

This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x6bc | Read/Write: R/W | Reset: 0x80008041 (0b1xxxxxxxxxxxxxxx1xxxxxxx01xxx0x1)

Bit	Reset	Description
31	ENABLE	ARB_OUTSTANDING_COUNT_ABOVE_SMMU: If enabled, arb_outstanding count includes requests in SMMU. If disabled, only count requests downstream of ring0. 0 = DISABLE 1 = ENABLE
15	ENABLE	RING0_THROTTLE_MASK_RING1_OUTPUT: If enabled, include the ring1 output (that is, all ring1 requests) in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
7	DISABLE	RING0_THROTTLE_MASK_RING1: if enabled, include ring1 in the ring0 meta-client group for throttling 0 = DISABLE 1 = ENABLE
6	ENABLE	RING0_THROTTLE_MASK_FTOP: if enabled, include ftop in the ring0 meta-client group for throttling 0 = DISABLE 1 = ENABLE
2	DISABLE	RING0_THROTTLE_MASK_PTC: If enabled, include ptc in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE
0	ENABLE	RING0_THROTTLE_MASK_MPCORER: If enabled, include mpcorer in the ring0 meta-client group for throttling. 0 = DISABLE 1 = ENABLE

18.11.1.109 MC_EMEM_ARB_TIMING_RFCPB_0

External Memory Arbitration Configuration: DRAM Timing : tRFCPB

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.
- This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x6c0 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxx001111111)

Bit	Reset	Description
8:0	0x7f	RFCPB: [PMC] specifies the per-bank auto refresh cycle time. This is the minimum number of cycles between a per-bank auto refresh command and a subsequent per-bank refresh (any bank) or activate command to that bank. Program to ceil(EMC.RFC_0_RFCPB/DIV)

18.11.1.110 MC_EMEM_ARB_TIMING_CCDMW_0

External Memory Arbitration Configuration: DRAM Timing : tCCDMW

> ***Note:*** *Although this chapter makes references to other DRAM types (such as DDR3L) only LPDDR3 and LPDDR4 memory types are officially supported and tested by NVIDIA. Other memory types should not be considered without first consulting NVIDIA to assess feasibility.*

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.
- This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x6c4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000001)

Bit	Reset	Description
5:0	0x1	CCDMW: [PMC] This field supports LPDDR4 DRAMs. It specifies the DRAM CAS to CAS delay timing when the current command is any type of write and the next command is a masked write to the same bank. If the DRAM type does not specify a timing parameter for tCCDMW (as in type DDR3L or LPDDR3), then set this field to tCCD timing (i.e., tCCD = 8 tCK at BL=16). Program to ceil(EMC.CCDMW/DIV)-1+SFA.

18.11.1.111 MC_EMEM_ARB_REFPB_HP_CTRL_0

External Memory Arbitration Configuration: REFPB - Per Bank Refresh High Priority control

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.
- This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x6f0 | Read/Write: R/W | Reset: 0x000a1020 (0bxxxxxxxxxx0001010x0010000x0100000)

Bit	Reset	Description
22:16	0xa	REFPB_OPEN_WORK_THRESH: [PMC] If the number of requests to open pages is greater than or equal to this value, a per-bank refresh is favored to be issued in lieu of an activate on the refresh bank as long as the activate is not for a high-priority expired request.
14:8	0x10	REFPB_THRESH_DISABLE_HP: [PMC] Per-bank refreshes become low-priority when the number of pending REFpb requests are less than or equal to this value.
6:0	0x20	REFPB_THRESH_ENABLE_HP: [PMC] Per-bank refreshes become high-priority when the number of pending REFpb requests are greater than or equal to this value.

18.11.1.112 MC_EMEM_ARB_REFPB_BANK_CTRL_0

External Memory Arbitration Configuration: REFPB - Bank control

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.
- This register is shadowed: see usage note at the top of *Section 18.11.1: MC Registers*.

Offset: 0x6f4 | Read/Write: R/W | Reset: 0x00002636 (0b0xxxxxxxxxxxxxxxx0100110x0110110)

Bit	Reset	Description
31	0x0	REFPB_EXPLICIT_BANK_MODE: [PMC] When set, REFpb cannot be issued simultaneously to more than one rank (i.e., device) of DRAM unless the next bank to be refreshed for both ranks are coincidentally the same (i.e., this is required for LPDDR4 support).
14:8	0x26	REFPB_THRESH_DISABLE_FORCE_CLOSE: [PMC] Per-bank refreshes will force bank closure when the number of pending REFpb requests are less than or equal to this value.
6:0	0x36	REFPB_THRESH_ENABLE_FORCE_CLOSE: [PMC] Per-bank refreshes will force bank closure when the number of pending REFpb requests are greater than or equal to this value.

18.11.1.113 MC_EMEM_ARB_OVERRIDE_1_0

Boot requirements:

- Some fields of register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0x968 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxx000000)

Bit	Reset	Description
31:12	0x0	EMEM_ARB_OVERRIDE_RESERVED
11	DISABLE	MULTI_BANK_REPLAY_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
5	DISABLE	EXPIRE_UPDATE_DEADLOCK_OVERRIDE: 0 = DISABLE 1 = ENABLE 1 = OVERRIDE
4:2	DISABLE	CPU_READ_PAGE_OPEN_POLICY: 0 = DISABLE 1 = ENABLE_ALL 2 = ENABLE_EACK_ACTIVE 3 = ENABLE_EARLY_ACK_ENABLE 4 = ENABLE_PO_HINT

Bit	Reset	Description
1:0	DISABLE	CPU_WRITE_PAGE_OPEN_POLICY : 0 = DISABLE 1 = ENABLE_ALL 2 = ENABLE_EACK_HINT

18.11.1.114 MC_CLIENT_HOTRESET_CTRL_1_0

Memory Client Hot Reset Control Register

Writing FLUSH_ENABLE to a bit in this register causes a flush to be performed on all clients for the selected swname. The status of the flush (done/in progress) can be monitored in the CLIENT_HOTRESET_STATUS register.

A proper client reset sequence is as follows:

- Set the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to block further access by the client, and start the flush process.
- Poll the CLIENT_HOTRESET_STATUS register until the appropriate bit returns FLUSH_DONE.
- Clear module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to reset the module.
- Set module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to release the module reset.
- Clear the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to allow transactions to flow.

Offset: 0x970 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx000xx0000xx000)

Bit	Reset	Description
13	DISABLE	TSECB_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
12	DISABLE	ETR_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
11	DISABLE	AXIAP_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
8	DISABLE	NVJPG_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
7	DISABLE	SE_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
6	DISABLE	APE_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
5	DISABLE	NVDEC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

2	DISABLE	GPU_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
1	DISABLE	ISP2B_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
0	DISABLE	SDMMC4A_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

18.11.1.115 MC_CLIENT_HOTRESET_STATUS_1_0

Memory Client Hot Reset Status Register

Contains one bit for each swname indicating the status of any flush that has been requested in the CLIENT_HOTRESET_CTRL register.

Note: *If no flush has been requested, this register returns FLUSH_DONE.*

Offset: 0x974 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13	X	TSECB_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
12	X	ETR_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
11	X	AXIAP_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
8	X	NVJPG_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
7	X	SE_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
6	X	APE_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
5	X	NVDEC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
2	X	GPU_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
1	X	ISP2B_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
0	X	SDMMC4A_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS

18.11.1.116 MC_VIDEO_PROTECT_GPU_OVERRIDE_0_0

Offset: 0x984 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VIDEO_PROTECT_GPU_OVERRIDE_0

18.11.1.117 MC_VIDEO_PROTECT_GPU_OVERRIDE_1_0

Offset: 0x988 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	VIDEO_PROTECT_GPU_OVERRIDE_1

18.11.1.118 MC_MTS_CARVEOUT_BOM_0

Offset: 0x9a0 | Read/Write: R/W | Reset: 0xfff00000 (0b111111111111xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:20	0xfff	MTS_CARVEOUT_BOM

18.11.1.119 MC_MTS_CARVEOUT_SIZE_MB_0

Offset: 0x9a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	MTS_CARVEOUT_SIZE_MB

18.11.1.120 MC_MTS_CARVEOUT_ADR_HI_0

Offset: 0x9a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	MTS_CARVEOUT_BOM_HI

18.11.1.121 MC_MTS_CARVEOUT_REG_CTRL_0

Offset: 0x9ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	ENABLED	MTS_CARVEOUT_WRITE_ACCESS 0 = ENABLED 1 = DISABLED

18.11.1.122 MC_SMMU_PTC_FLUSH_1_0

Program the higher address bits above PTC_FLUSH_ADR[31] in this field to flush a PDE or PTE group at a certain physical address.

Note: *Only a 10 GB physical address is supported.*

Page Table Cache Flush Address (Upper) Register

Offset: 0x9b8 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	0x0	PTC_FLUSH_ADR_HI: Physical address of PTE group to match for address flushes, PA[39:32]

18.11.1.123 MC_SECURITY_CFG3_0

Secure/Carveout Region Configuration: Base Address Higher Bits

Security Base:

- Can be only accessed by TrustZone-secured accesses from the secure clients.
- Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

- This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0x9bc | Read/Write: R/W | Secure Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	SECURITY_BOM_HI: SECURITY_BOM_HI has the higher address bits beyond 32 bits of the base of the secured region, limited to MB granularity.

18.11.1.124 MC_EMEM_BANK_SWIZZLE_CFG0_0

Bank Swizzling Register

Offset: 0x9c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EMEM_BANK_MASK0

18.11.1.125 MC_EMEM_BANK_SWIZZLE_CFG1_0

Bank Swizzling Register

Offset: 0x9c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EMEM_BANK_MASK1

18.11.1.126 MC_EMEM_BANK_SWIZZLE_CFG2_0

Bank Swizzling Register

Offset: 0x9c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EMEM_BANK_MASK2

18.11.1.127 MC_EMEM_BANK_SWIZZLE_CFG3_0

Bank Swizzling Register

Offset: 0x9cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	COL_NONE	EMEM_BANK_SWIZCOL 0 = COL_NONE 1 = COL_1KB 2 = COL_2KB 3 = COL_2KB_OFFSET_2 4 = COL_2KB_OFFSET_4

18.11.1.128 MC_SEC_CARVEOUT_ADR_HI_0

Offset: 0x9d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	SEC_CARVEOUT_BOM_HI: [PMC_SECURE] Higher address bits beyond 32 bits of byte-aligned address of the base address of the SEC carveout space.

18.11.1.129 MC_SMMU_DC1_ASID_0

SMMU DC1 ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xa88 | Read/Write: R/W | Secure Trust Zone Protected | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	DC1_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
30:24	0x0	DC1_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	DC1_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	DC1_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	DC1_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.130 MC_SMMU_SDMMC1A_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC1A ASID Assignment Register

Offset: 0xa94 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	SDMMC1A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
30:24	0x0	SDMMC1A_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	SDMMC1A_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	SDMMC1A_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	SDMMC1A_ASID: If translation enabled, ASID to use when va[33:32] = 0x0.

18.11.1.131 MC_SMMU_SDMMC2A_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC2A ASID Assignment Register

Offset: 0xa98 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	SDMMC2A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
30:24	0x0	SDMMC2A_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	SDMMC2A_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	SDMMC2A_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	SDMMC2A_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.132 MC_SMMU_SDMMC3A_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC3A ASID Assignment Register

Offset: 0xa9c | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	SDMMC3A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
30:24	0x0	SDMMC3A_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	SDMMC3A_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	SDMMC3A_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	SDMMC3A_ASID: If translation enabled, ASID to use when va[33:32] = 0x0.

18.11.1.133 MC_SMMU_SDMMC4A_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU SDMMC4A ASID Assignment Register

Offset: 0xaa0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	SDMMC4A_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30:24	0x0	SDMMC4A_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	SDMMC4A_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	SDMMC4A_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	SDMMC4A_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.134 MC_SMMU_ISP2B_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU ISP2B ASID Assignment Register

Offset: 0xaa4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	ISP2B_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
30:24	0x0	ISP2B_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	ISP2B_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	ISP2B_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	ISP2B_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.135 MC_SMMU_GPU_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU GPU ASID Assignment Register

Offset: 0xaa8 | Read/Write: R/W | Reset: 0xX0000000 (0bx0000000x0000000x0000000x0000000)

Bit	R/W	Reset	Description
31	RO	X	GPU_SMMU_ENABLE: Hardcoded for GPU to DISABLE. 0 = DISABLE 1 = ENABLE
30:24	RW	0x0	GPU_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	RW	0x0	GPU_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	RW	0x0	GPU_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	RW	0x0	GPU_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.136 MC_SMMU_GPUB_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU GPUB ASID Assignment Register

Offset: 0xaac | Read/Write: R/W | Reset: 0xX0000000 (0bx0000000x0000000x0000000x0000000)

Bit	R/W	Reset	Description
31	RO	X	GPUB_SMMU_ENABLE: Hardcoded for GPUB to ENABLE 0 = DISABLE 1 = ENABLE
30:24	RW	0x0	GPUB_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	RW	0x0	GPUB_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	RW	0x0	GPUB_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	RW	0x0	GPUB_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.137 MC_SMMU_PPCS2_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

SMMU PPCS2 ASID Assignment Register

Offset: 0xab0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	PPCS2_SMMU_ENABLE: If set, translation is enabled for this client. 0 = DISABLE 1 = ENABLE
6:0	0x0	PPCS2_ASID: If translation enabled, ASID to use when va[33:32] = 0x0.

18.11.1.138 MC_SMMU_NVDEC_ASID_0

SMMU NVDEC ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xab4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	NVDEC_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	NVDEC_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.139 MC_SMMU_APE_ASID_0

SMMU APE ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xab8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	APE_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	APE_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.140 MC_SMMU_SE_ASID_0

SMMU SE ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xabc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	SE_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	SE_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.141 MC_SMMU_NVJPG_ASID_0

SMMU NVJPG ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xac0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	NVJPG_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	NVJPG_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.142 MC_SMMU_HC1_ASID_0

SMMU HC1 ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xac4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	HC1_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	HC1_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.143 MC_SMMU_SE1_ASID_0

SMMU SE1 ASID Assignment Register

Enables or disables SMMU translation for the module and, if enabled, sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xac8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
31	DISABLE	SE1_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
6:0	0x0	SE1_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.144 MC_SMMU_AXIAP_ASID_0

SMMU AXIAP ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xacc | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	AXIAP_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	AXIAP_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	AXIAP_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	AXIAP_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	AXIAP_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.145 MC_SMMU_ETR_ASID_0

SMMU ETR ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xad0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	ETR_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	ETR_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	ETR_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	ETR_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	ETR_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.146 MC_SMMU_TSECB_ASID_0

SMMU TSECB ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xad4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000x0000000x0000000x0000000)

Bit	Reset	Description
31	DISABLE	TSECB_SMMU_ENABLE: if set, translation is enabled for this client 0 = DISABLE 1 = ENABLE
30:24	0x0	TSECB_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	0x0	TSECB_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	0x0	TSECB_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	0x0	TSECB_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.147 MC_SMMU_TSEC1_ASID_0

SMMU TSEC1 ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xad8 | Read/Write: R/W | Reset: 0xX0000000 (0bx0000000x0000000x0000000x0000000)

Bit	R/W	Reset	Description
31	RO	X	TSEC1_SMMU_ENABLE: Hardcoded for TSEC1 to DISABLE 0 = DISABLE 1 = ENABLE
30:24	RW	0x0	TSEC1_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	RW	0x0	TSEC1_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	RW	0x0	TSEC1_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	RW	0x0	TSEC1_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.148 MC_SMMU_TSECB1_ASID_0

SMMU TSECB1 ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xadc | Read/Write: R/W | Reset: 0xX0000000 (0bx0000000x0000000x0000000x0000000)

Bit	R/W	Reset	Description
31	RO	X	TSECB1_SMMU_ENABLE: Hardcoded for TSECB1 to DISABLE 0 = DISABLE 1 = ENABLE
30:24	RW	0x0	TSECB1_ASID_3: If translation enabled, ASID to use when va[33:32] = 0x3
22:16	RW	0x0	TSECB1_ASID_2: If translation enabled, ASID to use when va[33:32] = 0x2
14:8	RW	0x0	TSECB1_ASID_1: If translation enabled, ASID to use when va[33:32] = 0x1
6:0	RW	0x0	TSECB1_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.149 MC_SMMU_NVDEC1_ASID_0

SMMU NVDEC1 ASID Assignment Register

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation. This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xae0 | Read/Write: R/W | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	R/W	Reset	Description
31	RO	X	NVDEC1_SMMU_ENABLE: Hardcoded for NVDEC1 to DISABLE 0 = DISABLE 1 = ENABLE
6:0	RW	0x0	NVDEC1_ASID: If translation enabled, ASID to use when va[33:32] = 0x0

18.11.1.150 MC_EMEM_ARB_NISO_THROTTLE_MASK_1_0

External Memory Arbitration Configuration: Highest Ring Input NISO Throttle Mask

This register is used to group partition clients in the highest level snap-arbiter ring into a "NISO meta-client" group for the purpose of throttling. This is to allow soft-ISO clients in this ring to meet their bandwidth needs.

To prevent non-ISO (NISO) clients from filling the MC row-sorter (which could prevent soft-ISO clients from getting bandwidth), NVIDIA has implemented selective throttling at the input of ring2. To do this, the EMEM_ARB_NISO_THROTTLE_MASK register is used to specify which partition clients get included into a "NISO meta-client" group. To implement throttling, a total number of outstanding transactions counter (e.g., only one counter exists in the whole "NV_MC_cif" to track this, and is shared by all throttling circuits) gets compared to a programmable maximum limit for this NISO meta-client group. If the number is greater than the maximum limit, then all clients included in the "NISO meta-client" group get throttled based on the EMEM_ARB_NISO_THROTTLE priv register. This NISO_THROTTLE priv register is used to specify some number of stall cycles that get inserted at the input to the ring after every request from any of the clients included in the NISO group.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xb80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000x)

Bit	Reset	Description
5	DISABLED	NISO_THROTTLE_MASK_DFD: if enabled, include DFD in the NISO meta-client group for throttling 0 = DISABLED 1 = ENABLED
4	DISABLED	NISO_THROTTLE_MASK_HDAPC: if enabled, include HDAPC in the NISO meta-client group for throttling 0 = DISABLED 1 = ENABLED
3	DISABLED	NISO_THROTTLE_MASK_SDM: if enabled, include SDM in the NISO meta-client group for throttling 0 = DISABLED 1 = ENABLED
2	DISABLED	NISO_THROTTLE_MASK_GK2: if enabled, include GK2 in the NISO meta-client group for throttling 0 = DISABLED 1 = ENABLED
1	DISABLED	NISO_THROTTLE_MASK_JPG: if enabled, include JPG in the NISO meta-client group for throttling 0 = DISABLED 1 = ENABLED

18.11.1.151 MC_EMEM_ARB_HYSTERESIS_4_0

Per-client hysteresis settings

Controls whether a client is considered to be a hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xb84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	DISABLE	HYST_GPUSWR2: client gpuswr2 is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
8	DISABLE	HYST_GPUSRD2: client gpusrd2 is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	DISABLE	HYST_TSECSWRB: client tsecswrb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
6	DISABLE	HYST_TSECSRDB: client tsecsrdb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
5	DISABLE	HYST_ETRW: client etrw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	DISABLE	HYST_ETRR: client etrr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	DISABLE	HYST_AXIAPW: client axiapw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	DISABLE	HYST_AXIAPR: client axiapr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
1	DISABLE	HYST_SESWR: client seswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	DISABLE	HYST_SESRD: client sesrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.152 MC_EMEM_ARB_ISOCHRONOUS_4_0

Per-client isochronous settings

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type. This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xb94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	DISABLE	ISO_GPUSWR2: client gpuswr2 is treated as an isochronous client 0 = DISABLE 1 = ENABLE
8	DISABLE	ISO_GPUSRD2: client gpusrd2 is treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	DISABLE	ISO_TSECSWRB: client tsecswrb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	DISABLE	ISO_TSECSRDB: client tsecsrdb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
5	DISABLE	ISO_ETRW: client etrw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	DISABLE	ISO_ETRR: client etrr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
3	DISABLE	ISO_AXIAPW: client axiapw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
2	DISABLE	ISO_AXIAPR: client axiapr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
1	DISABLE	ISO_SESWR: client seswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
0	DISABLE	ISO_SESRD: client sesrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE

18.11.1.153 MC_SMMU_TRANSLATION_ENABLE_4_0

Per-client SMMU translation enables. Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xb98 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000Xff (0bxxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	R/W	Reset	Description
9	RO	X	SMMU_GPUSWR2_ENABLE: GPU translation is handled by swid, so this is required to be enabled 0 = DISABLE 1 = ENABLE
8	RO	X	SMMU_GPUSRD2_ENABLE: GPU translation is handled by swid, so this is required to be enabled 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SMMU_TSECSWRB_ENABLE: enable client tsecswrb to be translated by SMMU 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SMMU_TSECSRDB_ENABLE: enable client tsecsrdb to be translated by SMMU 0 = DISABLE 1 = ENABLE
5	RW	ENABLE	SMMU_ETRW_ENABLE: enable client etrw to be translated by SMMU 0 = DISABLE 1 = ENABLE
4	RW	ENABLE	SMMU_ETRR_ENABLE: enable client etrr to be translated by SMMU 0 = DISABLE 1 = ENABLE
3	RW	ENABLE	SMMU_AXIAPW_ENABLE: enable client axiapw to be translated by SMMU 0 = DISABLE 1 = ENABLE
2	RW	ENABLE	SMMU_AXIAPR_ENABLE: enable client axiapr to be translated by SMMU 0 = DISABLE 1 = ENABLE
1	RW	ENABLE	SMMU_SESWR_ENABLE: enable client seswr to be translated by SMMU 0 = DISABLE 1 = ENABLE
0	RW	ENABLE	SMMU_SESRD_ENABLE: enable client sesrd to be translated by SMMU 0 = DISABLE 1 = ENABLE

18.11.1.154 MC_EMEM_ARB_DHYSTERESIS_0_0

Per-client dynamic hysteresis settings

Controls whether a client is considered to be a dynamic hysteresis client for EMEM arbitration.

If a client is NOT marked dynamic hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbb0 | Read/Write: R/W | Reset: 0x10000000 (0b0001xxxx000xxx0000xxxxxxx0000000)

Bit	Reset	Description
31	0x0	DHYST_SATAR: client satar is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
30	0x0	DHYST_PPCSAHBSLVR: client ppcsahbslvr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
29	0x0	DHYST_PPCSAHBDMAR: client ppcsahbdmar is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
28	ENABLE	DHYST_NVENCSRD: client nvencsrd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x0	DHYST_HOST1XR: client host1xr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
22	0x0	DHYST_HOST1XDMAR: client host1xdmar is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
21	0x0	DHYST_HDAR: client hdar is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
17	0x0	DHYST_DISPLAYHCB: client displayhcb is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
16	0x0	DHYST_DISPLAYHC: client displayhc is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
15	0x0	DHYST_AVPCARM7R: client avpcarm7r is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
14	0x0	DHYST_AFIR: client afir is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
6	0x0	DHYST_DISPLAY0CB: client display0cb is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
5	0x0	DHYST_DISPLAY0C: client display0c is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
4	0x0	DHYST_DISPLAY0BB: client display0bb is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
3	0x0	DHYST_DISPLAY0B: client display0b is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
2	0x0	DHYST_DISPLAY0AB: client display0ab is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
1	0x0	DHYST_DISPLAY0A: client display0a is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
0	0x0	DHYST_PTCR: client ptcr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.155 MC_EMEM_ARB_DHYSTERESIS_1_0

Per-client dynamic hysteresis settings

Controls whether a client is considered to be a dynamic hysteresis client for EMEM arbitration. If a client is NOT marked dynamic hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbb4 | Read/Write: R/W | Reset: 0x00000800 (0bxx000x0xx00xx00xxxxx1xxx0xxxxxxx)

Bit	Reset	Description
29	0x0	DHYST_SATAW: client sataw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	DHYST_PPCSAHBSLVW: client ppcsahbslvw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
27	0x0	DHYST_PPCSAHBDMAW: client ppcsahbdmaw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
25	0x0	DHYST_MPCOREW: client mpcorew is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
22	0x0	DHYST_HOST1XW: client host1xw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
21	0x0	DHYST_HDAW: client hdaw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
18	0x0	DHYST_AVPCARM7W: client avpcarm7w is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
17	0x0	DHYST_AFIW: client afiw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
11	ENABLE	DHYST_NVENCSWR: client nvencswr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
7	0x0	DHYST_MPCORER: client mpcorer is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.156 MC_EMEM_ARB_DHYSTERESIS_2_0

Per-client dynamic hysteresis settings

Controls whether a client is considered to be a dynamic hysteresis client for EMEM arbitration. If a client is NOT marked dynamic hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbb8 | Read/Write: R/W | Reset: 0x030340d0 (0bxxxxx0110000xx11x10000xx11x1xxxx)

Bit	Reset	Description
26	0x0	DHYST_DISPLAYT: client displayt is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
25	ENABLE	DHYST_GPUSWR: client gpuswr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
24	ENABLE	DHYST_GPUSRD: client gpusrd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
23	0x0	DHYST_A9AVPSCW: client a9avpscw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
22	0x0	DHYST_A9AVPSCR: client a9avpscr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
21	0x0	DHYST_TSECSWR: client tsecswr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	DHYST_TSECSRD: client tsecsrd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
17	0x1	DHYST_ISPWBB: client ispwbb is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
16	0x1	DHYST_ISPWAB: client ispwab is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
14	0x1	DHYST_ISPRAB: client isprab is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
13	0x0	DHYST_XUSB_DEVW: client xusb_devw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
12	0x0	DHYST_XUSB_DEVR: client xusb_devr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
11	0x0	DHYST_XUSB_HOSTW: client xusb_hostw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
10	0x0	DHYST_XUSB_HOSTR: client xusb_hostr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
7	0x1	DHYST_ISPWB: client ispwb is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
6	0x1	DHYST_ISPWA: client ispwa is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
4	0x1	DHYST_ISPRA: client ispra is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.157 MC_EMEM_ARB_DHYSTERESIS_3_0

Per-client dynamic hysteresis settings

Controls whether a client is considered to be a dynamic hysteresis client for EMEM arbitration. If a client is NOT marked dynamic hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbbc | Read/Write: R/W | Reset: 0xc3043000 (0b11xx0011xxxx01xxxx11xxxx00000000)

Bit	Reset	Description
31	ENABLE	DHYST_NVJPGSWR: client nvjpgswr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
30	ENABLE	DHYST_NVJPGSRD: client nvjpgsrd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
27	0x0	DHYST_APEW: client apew is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
26	0x0	DHYST_APER: client aper is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	ENABLE	DHYST_NVDECSWR: client nvdecswr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
24	ENABLE	DHYST_NVDECSRD: client nvdecsrd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
19	0x0	DHYST_DISPLAYD: client displayd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
18	ENABLE	DHYST_VIW: client viw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
13	ENABLE	DHYST_VICSWR: client vicswr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
12	ENABLE	DHYST_VICSRD: client vicsrd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
7	0x0	DHYST_SDMMCWAB: client sdmmcwab is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
6	0x0	DHYST_SDMMCW: client sdmmcw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
5	0x0	DHYST_SDMMCWAA: client sdmmcwaa is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
4	0x0	DHYST_SDMMCWA: client sdmmcwa is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
3	0x0	DHYST_SDMMCRAB: client sdmmcrab is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
2	0x0	DHYST_SDMMCR: client sdmmcr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
1	0x0	DHYST_SDMMCRAA: client sdmmcraa is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
0	0x0	DHYST_SDMMCRA: client sdmmcra is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.158 MC_EMEM_ARB_DHYSTERESIS_4_0

Per-client dynamic hysteresis settings

Controls whether a client is considered to be a dynamic hysteresis client for EMEM arbitration. If a client is NOT marked dynamic hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbc0 | Read/Write: R/W | Reset: 0x00000300 (0bxxxxxxxxxxxxxxxxxxxxxxxx1100000000)

Bit	Reset	Description
9	ENABLE	DHYST_GPUSWR2: client gpuswr2 is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	ENABLE	DHYST_GPUSRD2: client gpusrd2 is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
7	0x0	DHYST_TSECSWRB: client tsecswrb is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
6	0x0	DHYST_TSECSRDB: client tsecsrdb is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
5	0x0	DHYST_ETRW: client etrw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
4	0x0	DHYST_ETRR: client etrr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
3	0x0	DHYST_AXIAPW: client axiapw is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
2	0x0	DHYST_AXIAPR: client axiapr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
1	0x0	DHYST_SESWR: client seswr is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE
0	0x0	DHYST_SESRD: client sesrd is treated as a dynamic hysteresis client 0 = DISABLE 1 = ENABLE

18.11.1.159 MC_EMEM_ARB_DHYST_CTRL_0

Control register for dynamic hysteresis logic (power saving feature)

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbcc | Read/Write: R/W | Reset: 0x00000002 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx10)

Bit	Reset	Description
31	DISABLE	DHYST_ENABLE: The dynamic hysteresis logic will not be enabled unless this bit is set. When disabled, the dynamic hysteresis logic is forced quiescent (to save power) 0 = DISABLE 1 = ENABLE
1:0	0x2	DHYST_BW_MON_INTERVAL: Specifies the bandwidth monitor interval to be used for dynamic hysteresis. 0 = 8 mclk interval 1 = 16 mclk interval 2 = 32 mclk interval (default) 3 = 64 mclk interval In general, larger values of this interval should save more power, but at the potential cost of memory latency due to longer ts2aa_holdoff exit response time.

18.11.1.160 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_0_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 0% and 12.5%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbd0 | Read/Write: R/W | Reset: 0x00000100 (0bxxxxxxxxxxxxxxxxxxxxxx000100000000)

Bit	Reset	Description
11:0	0x100	DHYST_DELAY_BIN_0

18.11.1.161 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_1_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 12.5% and 25%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbd4 | Read/Write: R/W | Reset: 0x000000a0 (0bxxxxxxxxxxxxxxxxxxxxxx000010100000)

Bit	Reset	Description
11:0	0xa0	DHYST_DELAY_BIN_1

18.11.1.162 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_2_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 25% and 37.5%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbd8 | Read/Write: R/W | Reset: 0x00000060 (0bxxxxxxxxxxxxxxxxxxxxxx000001100000)

Bit	Reset	Description
11:0	0x60	DHYST_DELAY_BIN_2

18.11.1.163 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_3_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 37.5% and 50%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbdc | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxx000000110100)

Bit	Reset	Description
11:0	0x34	DHYST_DELAY_BIN_3

18.11.1.164 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_4_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 50% and 62.5%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbe0 | Read/Write: R/W | Reset: 0x00000018 (0bxxxxxxxxxxxxxxxxxxxxxx000000011000)

Bit	Reset	Description
11:0	0x18	DHYST_DELAY_BIN_4

18.11.1.165 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_5_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 62.5% and 75%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbe4 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxx000000001100)

Bit	Reset	Description
11:0	0xc	DHYST_DELAY_BIN_5

18.11.1.166 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_6_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 75% and 87.5%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbe8 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxx000000000100)

Bit	Reset	Description
11:0	0x4	DHYST_DELAY_BIN_6

18.11.1.167 MC_EMEM_ARB_DHYST_TIMEOUT_UTIL_7_0

Controls the timeout delay (in mcclk's) for dynamic hysteresis holdoff when DRAM utilization is between 87.5% and 100%.

Boot requirements:

- This register should be saved to SDRAM and restored by the OS during warm boot.

Offset: 0xbec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	DHYST_DELAY_BIN_7

18.11.1.168 MC_DA_CONFIG0_0

This register is used to select which scheme of arbitration the DA uses.

Offset: 0x9dc I Read/Write: R/W I Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	ENABLE	NEW_ARB_SCHEME:[PMC] 0 = DISABLE 1 = ENABLE

18.11.1.169 MC_AHB_PTSA_MIN_0

DDA minimum value for direct client ahb PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4e0 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_AHB

18.11.1.170 MC_AUD_PTSA_MIN_0

DDA minimum value for direct client aud PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x54c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_AUD

18.11.1.171 MC_MLL_MPCORER_PTSA_RATE_0

DDA rate for mll_mpcorer PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x44c I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_MLL_MPCORER

18.11.1.172 MC_RING2_PTSA_RATE_0

DDA rate for ring2 PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x440 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_RING2

18.11.1.173 MC_USBD_PTSA_RATE_0

DDA rate for usbd PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x530 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_USBD

18.11.1.174 MC_USBX_PTSA_MIN_0

DDA minimum value for direct client usbx PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x528 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_USBX

18.11.1.175 MC_USBD_PTSA_MIN_0

DDA minimum value for direct client usbd PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x534 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_USBD

18.11.1.176 MC_APB_PTSA_MAX_0

DDA maximum value for direct client apb PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4f0 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_APB

18.11.1.177 MC_JPG_PTSA_RATE_0

DDA rate for jpg PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x584 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_JPG

18.11.1.178 MC_DIS_PTSA_MIN_0

DDA minimum value for direct client dis PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x420 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_DIS

18.11.1.179 MC_AVP_PTSA_MAX_0

DDA maximum value for direct client avp PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4fc I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_AVP

18.11.1.180 MC_AVP_PTSA_RATE_0

DDA rate for avp PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4f4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_AVP

18.11.1.181 MC_RING1_PTSA_MIN_0

DDA minimum value for direct client ring1 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x480 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_RING1

18.11.1.182 MC_DIS_PTSA_MAX_0

DDA maximum value for direct client dis PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x424 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_DIS

18.11.1.183 MC_SD_PTSA_MAX_0

DDA maximum value for direct client sd PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4d8 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_SD

18.11.1.184 MC_MSE_PTSA_RATE_0

DDA rate for mse PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4c4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_MSE

18.11.1.185 MC_VICPC_PTSA_MIN_0

DDA minimum value for direct client vicpc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x558 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_VICPC

18.11.1.186 MC_PCX_PTSA_MAX_0

DDA maximum value for direct client pcx PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4b4 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_PCX

18.11.1.187 MC_ISP_PTSA_RATE_0

DDA rate for isp PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4a0 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_ISP

18.11.1.188 MC_A9AVPPC_PTSA_MIN_0

DDA minimum value for direct client a9avppc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x48c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_A9AVPPC

18.11.1.189 MC_RING2_PTSA_MAX_0

DDA maximum value for direct client ring2 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x448 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_RING2

18.11.1.190 MC_AUD_PTSA_RATE_0

DDA rate for aud PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x548 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_AUD

18.11.1.191 MC_HOST_PTSA_MIN_0

DDA minimum value for direct client host PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x51c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_HOST

18.11.1.192 MC_MLL_MPCORER_PTSA_MAX_0

DDA maximum value for direct client mll_mpcorer PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x454 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_MLL_MPCORER

18.11.1.193 MC_SD_PTSA_MIN_0

DDA minimum value for direct client sd PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4d4 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_SD

18.11.1.194 MC_RING1_PTSA_RATE_0

DDA rate for ring1 PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x47c I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_RING1

18.11.1.195 MC_JPG_PTSA_MIN_0

DDA minimum value for direct client jpg PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x588 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_JPG

18.11.1.196 MC_HDAPC_PTSA_MIN_0

DDA minimum value for direct client hdapc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x62c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_HDAPC

18.11.1.197 MC_AVP_PTSA_MIN_0

DDA minimum value for direct client avp PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4f8 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_AVP

18.11.1.198 MC_JPG_PTSA_MAX_0

DDA maximum value for direct client jpg PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x58c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_JPG

18.11.1.199 MC_VE_PTSA_MAX_0

DDA maximum value for direct client ve PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x43c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_VE

18.11.1.200 MC_DFD_PTSA_MAX_0

DDA maximum value for direct client dfd PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x63c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_DFD

18.11.1.201 MC_VICPC_PTSA_RATE_0

DDA rate for vicpc PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x554 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_VICPC

18.11.1.202 MC_GK_PTSA_MAX_0

DDA maximum value for direct client gk PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x544 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_GK

18.11.1.203 MC_VICPC_PTSA_MAX_0

DDA maximum value for direct client vicpc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x55c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_VICPC

18.11.1.204 MC_SDM_PTSA_MAX_0

DDA maximum value for direct client sdm PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x624 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_SDM

18.11.1.205 MC_SAX_PTSA_RATE_0

DDA rate for sax PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4b8 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_SAX

18.11.1.206 MC_PCX_PTSA_MIN_0

DDA minimum value for direct client pcx PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4b0 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_PCX

18.11.1.207 MC_APB_PTSA_MIN_0

DDA minimum value for direct client apb PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4ec I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_APB

18.11.1.208 MC_GK2_PTSA_MIN_0

DDA minimum value for direct client gk2 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x614 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_GK2

18.11.1.209 MC_PCX_PTSA_RATE_0

DDA rate for pcx PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4ac I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_PCX

18.11.1.210 MC_RING1_PTSA_MAX_0

DDA maximum value for direct client ring1 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x484 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_RING1

18.11.1.211 MC_HDAPC_PTSA_RATE_0

DDA rate for hdapc PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x628 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_HDAPC

18.11.1.212 MC_MLL_MPCORER_PTSA_MIN_0

DDA minimum value for direct client mll_mpcorer PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x450 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_MLL_MPCORER

18.11.1.213 MC_GK2_PTSA_MAX_0

DDA maximum value for direct client gk2 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x618 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_GK2

18.11.1.214 MC_AUD_PTSA_MAX_0

DDA maximum value for direct client aud PTSA.

Indicate negative values by their 2's complement

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x550 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_AUD

18.11.1.215 MC_GK2_PTSA_RATE_0

DDA rate for gk2 PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x610 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_GK2

18.11.1.216 MC_ISP_PTSA_MAX_0

DDA maximum value for direct client isp PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4a8 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_ISP

18.11.1.217 MC_DISB_PTSA_RATE_0

DDA rate for disb PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x428 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_DISB

18.11.1.218 MC_VE2_PTSA_MAX_0

DDA maximum value for direct client ve2 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x49c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_VE2

18.11.1.219 MC_DFD_PTSA_MIN_0

DDA minimum value for direct client dfd PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x638 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_DFD

18.11.1.220 MC_FTOP_PTSA_RATE_0

DDA rate for ftop PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x50c I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_FTOP

18.11.1.221 MC_A9AVPPC_PTSA_RATE_0

DDA rate for a9avppc PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x488 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_A9AVPPC

18.11.1.222 MC_VE2_PTSA_MIN_0

DDA minimum value for direct client ve2 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x498 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_VE2

18.11.1.223 MC_USBX_PTSA_MAX_0

DDA maximum value for direct client usbx PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x52c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_USBX

18.11.1.224 MC_DIS_PTSA_RATE_0

DDA rate for dis PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x41c I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_DIS

18.11.1.225 MC_USBD_PTSA_MAX_0

DDA maximum value for direct client usbd PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x538 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_USBD

18.11.1.226 MC_A9AVPPC_PTSA_MAX_0

DDA maximum value for direct client a9avppc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x490 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_A9AVPPC

18.11.1.227 MC_USBX_PTSA_RATE_0

DDA rate for usbx PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x524 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_USBX

18.11.1.228 MC_FTOP_PTSA_MAX_0

DDA maximum value for direct client ftop PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x514 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_FTOP

18.11.1.229 MC_HDAPC_PTSA_MAX_0

DDA maximum value for direct client hdapc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x630 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_HDAPC

18.11.1.230 MC_SD_PTSA_RATE_0

DDA rate for sd PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4d0 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_SD

18.11.1.231 MC_DFD_PTSA_RATE_0

DDA rate for dfd PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x634 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_DFD

18.11.1.232 MC_FTOP_PTSA_MIN_0

DDA minimum value for direct client ftop PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x510 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_FTOP

18.11.1.233 MC_SDM_PTSA_RATE_0

DDA rate for sdm PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x61c I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_SDM

18.11.1.234 MC_AHB_PTSA_RATE_0

DDA rate for ahb PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4dc I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_AHB

18.11.1.235 MC_SMMU_SMMU_PTSA_MAX_0

DDA maximum value for direct client ptc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x460 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_SMMU_SMMU

18.11.1.236 MC_RING2_PTSA_MIN_0

DDA minimum value for direct client ring2 PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x444 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_RING2

18.11.1.237 MC_SDM_PTSA_MIN_0

DDA minimum value for direct client sdm PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x620 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_SDM

18.11.1.238 MC_APB_PTSA_RATE_0

DDA rate for APB PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4e8 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_APB

18.11.1.239 MC_MSE_PTSA_MIN_0

DDA minimum value for direct client mse PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4c8 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_MSE

18.11.1.240 MC_HOST_PTSA_RATE_0

DDA rate for host PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x518 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_HOST

18.11.1.241 MC_VE_PTSA_RATE_0

DDA rate for ve PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x434 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_VE

18.11.1.242 MC_AHB_PTSA_MAX_0

DDA maximum value for direct client ahb PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4e4 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_AHB

18.11.1.243 MC_SAX_PTSA_MIN_0

DDA minimum value for direct client sax PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4bc I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_SAX

18.11.1.244 MC_SMMU_SMMU_PTSA_MIN_0

DDA minimum value for direct client ptc PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x45c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_SMMU_SMMU

18.11.1.245 MC_ISP_PTSA_MIN_0

DDA minimum value for direct client isp PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4a4 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_ISP

18.11.1.246 MC_HOST_PTSA_MAX_0

DDA maximum value for direct client host PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x520 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_HOST

18.11.1.247 MC_SAX_PTSA_MAX_0

DDA maximum value for direct client sax PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4c0 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_SAX

18.11.1.248 MC_VE_PTSA_MIN_0

DDA minimum value for direct client ve PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x438 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_VE

18.11.1.249 MC_GK_PTSA_MIN_0

DDA minimum value for direct client gk PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x540 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_GK

18.11.1.250 MC_MSE_PTSA_MAX_0

DDA maximum value for direct client mse PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x4cc I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_MSE

18.11.1.251 MC_DISB_PTSA_MAX_0

DDA maximum value for direct client disb PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x430 I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MAX_DISB

18.11.1.252 MC_DISB_PTSA_MIN_0

DDA minimum value for direct client disb PTSA.

Indicate negative values by their 2's complement.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x42c I Read/Write: R/W I Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	PTSA_MIN_DISB

18.11.1.253 MC_SMMU_SMMU_PTSA_RATE_0

DDA rate for ptc PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x458 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_SMMU_SMMU

18.11.1.254 MC_VE2_PTSA_RATE_0

DDA rate for ve2 PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x494 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_VE2

18.11.1.255 MC_GK_PTSA_RATE_0

DDA rate for gk PTSA

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x53c I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PTSA_RATE_GK

18.11.1.256 MC_PTSA_GRANT_DECREMENT_0

The amount that the PTSA DDAs will be decremented each cycle that there is a grant should be programmed to the fraction of total memory BW / maximum total memory BW

PTSA Grand Decrement

Boot requirements: This register should be saved in the SDRAM and restored by the OS during warmboot.

This register is shadowed: see usage note at top of *Section 18.11.1: MC Registers*.

Offset: 0x960 I Read/Write: R/W I Reset: 0x00001000 (0bxxxxxxxxxxxxxxxxxxxx1000000000000)

Bit	Reset	Description
12	0x1	PTSA_GRANT_DECREMENT_INT: integer portion (0 or 1)
11:0	0x0	PTSA_GRANT_DECREMENT_FRAC: fractional portion

18.11.1.257 MC_LATENCY_ALLOWANCE_AVPC_0_0

Latency allowance settings for AVPC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2e4 I Read/Write: R/W I Reset: 0x00800004 (0bxxxxxxxx10000000xxxxxxxx00000100)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_AVPCARM7W
7:0	0x4	ALLOWANCE_AVPCARM7R

18.11.1.258 MC_LATENCY_ALLOWANCE_AXIAP_0_0

Latency allowance settings for AXIAP clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3a0 I Read/Write: R/W I Reset: 0x008000ff (0bxxxxxxxx10000000xxxxxxxx11111111)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_AXIAPW
7:0	0xff	ALLOWANCE_AXIAPR

18.11.1.259 MC_LATENCY_ALLOWANCE_XUSB_1_0

Latency allowance settings for XUSB_DEV clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x380 I Read/Write: R/W I Reset: 0x00800039 (0bxxxxxxxx10000000xxxxxxxx00111001)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_XUSB_DEVW
7:0	0x39	ALLOWANCE_XUSB_DEVR

18.11.1.260 MC_LATENCY_ALLOWANCE_ISP2B_0_0

Latency allowance settings for ISP2B clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x384 I Read/Write: R/W I Reset: 0x00000018 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00011000)

Bit	Reset	Description
7:0	0x18	ALLOWANCE_ISPRAB

18.11.1.261 MC_LATENCY_ALLOWANCE_SDMMCAA_0_0

Latency allowance settings for SDMMC2A clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3bc I Read/Write: R/W I Reset: 0x0080005a (0bxxxxxxxx10000000xxxxxxxx01011010)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_SDMMCWAA
7:0	0x5a	ALLOWANCE_SDMMCRAA

18.11.1.262 MC_LATENCY_ALLOWANCE_SDMMCA_0_0

Latency allowance settings for SDMMC1A clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3b8 I Read/Write: R/W I Reset: 0x00800049 (0bxxxxxxxx10000000xxxxxxxx01001001)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_SDMMCWA
7:0	0x49	ALLOWANCE_SDMMCRA

18.11.1.263 MC_LATENCY_ALLOWANCE_ISP2_0_0

Latency allowance settings for ISP2 clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x370 I Read/Write: R/W I Reset: 0x00000018 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00011000)

Bit	Reset	Description
7:0	0x18	ALLOWANCE_ISPRA

18.11.1.264 MC_LATENCY_ALLOWANCE_SE_0_0

Latency allowance settings for SE clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3e0 I Read/Write: R/W I Reset: 0x0080002e (0bxxxxxxxx10000000xxxxxxxx00101110)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_SESWR
7:0	0x2e	ALLOWANCE_SESRD

18.11.1.265 MC_LATENCY_ALLOWANCE_ISP2_1_0

Latency allowance settings for ISP2 clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x374 I Read/Write: R/W I Reset: 0x00800080 (0bxxxxxxxx10000000xxxxxxxx10000000)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_ISPWB
7:0	0x80	ALLOWANCE_ISPWA

18.11.1.266 MC_LATENCY_ALLOWANCE_DC_0_0

Latency allowance settings for DC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2e8 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	ALLOWANCE_DISPLAY0B
7:0	0x1e	ALLOWANCE_DISPLAY0A

18.11.1.267 MC_LATENCY_ALLOWANCE_VIC_0_0

Latency allowance settings for VIC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x394 I Read/Write: R/W I Reset: 0x0080001a (0bxxxxxxxx10000000xxxxxxxx00011010)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_VICSWR
7:0	0x1a	ALLOWANCE_VICSRD

18.11.1.268 MC_LATENCY_ALLOWANCE_DCB_1_0

Latency allowance settings for DCB clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2f8 I Read/Write: R/W I Reset: 0x0000001e (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00011110)

Bit	Reset	Description
7:0	0x1e	ALLOWANCE_DISPLAY0CB

18.11.1.269 MC_LATENCY_ALLOWANCE_NVDEC_0_0

Latency allowance settings for NVDEC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3d8 I Read/Write: R/W I Reset: 0x00800023 (0bxxxxxxxx10000000xxxxxxxx00100011)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_NVDECSWR
7:0	0x23	ALLOWANCE_NVDECSRD

18.11.1.270 MC_LATENCY_ALLOWANCE_DCB_2_0

Latency allowance settings for DCB clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2fc I Read/Write: R/W I Reset: 0x0000001e (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00011110)

Bit	Reset	Description
7:0	0x1e	ALLOWANCE_DISPLAYHCB

18.11.1.271 MC_LATENCY_ALLOWANCE_TSEC_0_0

Latency allowance settings for TSEC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x390 I Read/Write: R/W I Reset: 0x0080009b (0bxxxxxxxx10000000xxxxxxxx10011011)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_TSECSWR
7:0	0x9b	ALLOWANCE_TSECSRD

18.11.1.272 MC_LATENCY_ALLOWANCE_DC_2_0

Latency allowance settings for DC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2f0 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	ALLOWANCE_DISPLAYT
7:0	0x1e	ALLOWANCE_DISPLAYHC

18.11.1.273 MC_SCALED_LATENCY_ALLOWANCE_DISPLAY0AB_0

Scaled Latency Allowance settings for DISPLAY0AB

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x694 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	SCALED_2_ALLOWANCE_HI_DISPLAY0AB
7:0	0x1e	SCALED_2_ALLOWANCE_LO_DISPLAY0AB

18.11.1.274 MC_LATENCY_ALLOWANCE_PPCS_1_0

Latency allowance settings for PPCS clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x348 I Read/Write: R/W I Reset: 0x00800080 (0bxxxxxxxx10000000xxxxxxxx10000000)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_PPCSAHBSLVW
7:0	0x80	ALLOWANCE_PPCSAHBDMAW

18.11.1.275 MC_LATENCY_ALLOWANCE_XUSB_0_0

Latency allowance settings for XUSB_HOST clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x37c I Read/Write: R/W I Reset: 0x0080007a (0bxxxxxxxx10000000xxxxxxxx01111010)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_XUSB_HOSTW
7:0	0x7a	ALLOWANCE_XUSB_HOSTR

18.11.1.276 MC_LATENCY_ALLOWANCE_PPCS_0_0

Latency allowance settings for PPCS clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x344 I Read/Write: R/W I Reset: 0x001a0049 (0bxxxxxxxx00011010xxxxxxxx01001001)

Bit	Reset	Description
23:16	0x1a	ALLOWANCE_PPCSAHBSLVR
7:0	0x49	ALLOWANCE_PPCSAHBDMAR

18.11.1.277 MC_LATENCY_ALLOWANCE_TSECB_0_0

Latency allowance settings for TSECB clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3f0 I Read/Write: R/W I Reset: 0x0080009b (0bxxxxxxxx10000000xxxxxxxx10011011)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_TSECSWRB
7:0	0x9b	ALLOWANCE_TSECSRDB

18.11.1.278 MC_LATENCY_ALLOWANCE_AFI_0_0

Latency allowance settings for AFI clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2e0 I Read/Write: R/W I Reset: 0x0080002e (0bxxxxxxxx10000000xxxxxxxx00101110)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_AFIW
7:0	0x2e	ALLOWANCE_AFIR

18.11.1.279 MC_SCALED_LATENCY_ALLOWANCE_DISPLAY0B_0

Scaled Latency Allowance settings for DISPLAY0B

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x698 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	SCALED_2_ALLOWANCE_HI_DISPLAY0B
7:0	0x1e	SCALED_2_ALLOWANCE_LO_DISPLAY0B

18.11.1.280 MC_LATENCY_ALLOWANCE_DC_1_0

Latency allowance settings for DC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2ec I Read/Write: R/W I Reset: 0x0000001e (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00011110)

Bit	Reset	Description
7:0	0x1e	ALLOWANCE_DISPLAY0C

18.11.1.281 MC_LATENCY_ALLOWANCE_APE_0_0

Latency allowance settings for APE clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3dc I Read/Write: R/W I Reset: 0x008000ff (0bxxxxxxxx10000000xxxxxxxx11111111)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_APEW
7:0	0xff	ALLOWANCE_APER

18.11.1.282 MC_SCALED_LATENCY_ALLOWANCE_DISPLAY0C_0

Scaled Latency Allowance settings for DISPLAY0C

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x6a0 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	SCALED_2_ALLOWANCE_HI_DISPLAY0C
7:0	0x1e	SCALED_2_ALLOWANCE_LO_DISPLAY0C

18.11.1.283 MC_LATENCY_ALLOWANCE_A9AVP_0_0

Latency allowance settings for A9AVP clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3a4 I Read/Write: R/W I Reset: 0x00800004 (0bxxxxxxxx10000000xxxxxxxx00000100)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_A9AVPSCW
7:0	0x4	ALLOWANCE_A9AVPSCR

18.11.1.284 MC_LATENCY_ALLOWANCE_GPU2_0_0

Latency allowance settings for GPU clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'.This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3e8 I Read/Write: R/W I Reset: 0x0080001a (0bxxxxxxxx10000000xxxxxxxx00011010)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_GPUSWR2
7:0	0x1a	ALLOWANCE_GPUSRD2

18.11.1.285 MC_LATENCY_ALLOWANCE_DCB_0_0

Latency allowance settings for DCB clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x2f4 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	ALLOWANCE_DISPLAY0BB
7:0	0x1e	ALLOWANCE_DISPLAY0AB

18.11.1.286 MC_LATENCY_ALLOWANCE_HC_1_0

Latency allowance settings for HC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x314 I Read/Write: R/W I Reset: 0x00000080 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx10000000)

Bit	Reset	Description
7:0	0x80	ALLOWANCE_HOST1XW

18.11.1.287 MC_LATENCY_ALLOWANCE_SDMMC_0_0

Latency allowance settings for SDMMC3A clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3c0 I Read/Write: R/W I Reset: 0x00800049 (0bxxxxxxxx10000000xxxxxxxx01001001)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_SDMMCW
7:0	0x49	ALLOWANCE_SDMMCR

18.11.1.288 MC_LATENCY_ALLOWANCE_NVJPG_0_0

Latency allowance settings for NVJPG clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3e4 I Read/Write: R/W I Reset: 0x00800023 (0bxxxxxxxx10000000xxxxxxxx00100011)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_NVJPGSWR
7:0	0x23	ALLOWANCE_NVJPGSRD

18.11.1.289 MC_LATENCY_ALLOWANCE_PTC_0_0

Latency allowance settings for PTC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x34c I Read/Write: R/WI Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	ALLOWANCE_PTCR

18.11.1.290 MC_LATENCY_ALLOWANCE_ETR_0_0

Latency allowance settings for ETR clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3ec I Read/Write: R/W I Reset: 0x008000ff (0bxxxxxxxx10000000xxxxxxxx11111111)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_ETRW
7:0	0xff	ALLOWANCE_ETRR

18.11.1.291 MC_LATENCY_ALLOWANCE_MPCORE_0_0

Latency allowance settings for MPCORE clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x320 I Read/Write: R/W I Reset: 0x00800004 (0bxxxxxxxx10000000xxxxxxxx00000100)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_MPCOREW
7:0	0x4	ALLOWANCE_MPCORER

18.11.1.292 MC_LATENCY_ALLOWANCE_VI2_0_0

Latency allowance settings for VI clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x398 I Read/Write: R/W I Reset: 0x00000080 (0bxxxxxxxxxxxxxxxxxxxxxxxxx10000000)

Bit	Reset	Description
7:0	0x80	ALLOWANCE_VIW

18.11.1.293 MC_SCALED_LATENCY_ALLOWANCE_DISPLAY0BB_0

Scaled Latency Allowance settings for DISPLAY0BB

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x69c I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	SCALED_2_ALLOWANCE_HI_DISPLAY0BB
7:0	0x1e	SCALED_2_ALLOWANCE_LO_DISPLAY0BB

18.11.1.294 MC_SCALED_LATENCY_ALLOWANCE_DISPLAY0CB_0

Scaled Latency Allowance settings for DISPLAY0CB

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x6a4 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	SCALED_2_ALLOWANCE_HI_DISPLAY0CB
7:0	0x1e	SCALED_2_ALLOWANCE_LO_DISPLAY0CB

18.11.1.295 MC_LATENCY_ALLOWANCE_SATA_0_0

Latency allowance settings for SATA clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x350 I Read/Write: R/W I Reset: 0x00800065 (0bxxxxxxxx10000000xxxxxxxx01100101)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_SATAW
7:0	0x65	ALLOWANCE_SATAR

18.11.1.296 MC_SCALED_LATENCY_ALLOWANCE_DISPLAY0A_0

Scaled Latency Allowance settings for DISPLAY0A

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x690 I Read/Write: R/W I Reset: 0x001e001e (0bxxxxxxxxx00011110xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x1e	SCALED_2_ALLOWANCE_HI_DISPLAY0A
7:0	0x1e	SCALED_2_ALLOWANCE_LO_DISPLAY0A

18.11.1.297 MC_LATENCY_ALLOWANCE_HC_0_0

Latency allowance settings for HC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x310 I Read/Write: R/W I Reset: 0x0050001e (0bxxxxxxxx01010000xxxxxxxx00011110)

Bit	Reset	Description
23:16	0x50	ALLOWANCE_HOST1XR
7:0	0x1e	ALLOWANCE_HOST1XDMAR

18.11.1.298 MC_LATENCY_ALLOWANCE_DC_3_0

Latency allowance settings for DC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3c8 I Read/Write: R/W I Reset: 0x0000001e (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00011110)

Bit	Reset	Description
7:0	0x1e	ALLOWANCE_DISPLAYD

18.11.1.299 MC_LATENCY_ALLOWANCE_GPU_0_0

Latency allowance settings for GPU clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3ac I Read/Write: R/W I Reset: 0x0080001a (0bxxxxxxxx10000000xxxxxxxx00011010)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_GPUSWR
7:0	0x1a	ALLOWANCE_GPUSRD

18.11.1.300 MC_LATENCY_ALLOWANCE_SDMMCAB_0_0

Latency allowance settings for SDMMC4A clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x3c4 I Read/Write: R/W I Reset: 0x0080005a (0bxxxxxxxx10000000xxxxxxxx01011010)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_SDMMCWAB
7:0	0x5a	ALLOWANCE_SDMMCRAB

18.11.1.301 MC_LATENCY_ALLOWANCE_ISP2B_1_0

Latency allowance settings for ISP2B clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'. This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x388 I Read/Write: R/W I Reset: 0x00800080 (0bxxxxxxxx10000000xxxxxxxx10000000)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_ISPWBB
7:0	0x80	ALLOWANCE_ISPWAB

18.11.1.302 MC_LATENCY_ALLOWANCE_NVENC_0_0

Latency allowance settings for NVENC clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'.This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x328 I Read/Write: R/W I Reset: 0x00800023 (0bxxxxxxxx10000000xxxxxxxx00100011)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_NVENCSWR
7:0	0x23	ALLOWANCE_NVENCSRD

18.11.1.303 MC_LATENCY_ALLOWANCE_HDA_0_0

Latency allowance settings for HDA clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request. See EMEM_ARB_CFG for definition of the units 'ticks'.This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warmboot.

Offset: 0x318 I Read/Write: R/W I Reset: 0x00800024 (0bxxxxxxxx10000000xxxxxxxx00100100)

Bit	Reset	Description
23:16	0x80	ALLOWANCE_HDAW
7:0	0x24	ALLOWANCE_HDAR

18.11.2 EMC Registers

USAGE NOTE: Many EMC register fields are shadowed:

- Writes to shadowed register fields update the shadow copy (this is default, assumes DBG.WRITE_MUX==ASSEMBLY).
- Reads to shadowed register fields return the active copy (this is default, assumes DBG.READ_MUX==ACTIVE).

This allows a new set of frequency-dependent timing parameters to be written to the shadow registers while memory traffic is ongoing, then when the parameters are completely written, the EMC hardware can perform a timing-safe switch.

Such switches can be triggered via two methods: TIMING_CONTROL.TIMING_UPDATE (generally used during initialization), or the automatic CAR/EMC handshake on a clock-frequency or divider change (assuming CFG_2.CLKCHANGE_REQ_ENABLE==ENABLED).

Registers that are shadowed are marked by a comment:

> ***Note:*** *This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.*

Occasionally, only certain fields in the register will be shadowed; if so they are noted after the above comment.

USAGE NOTE: Many EMC registers play crucial roles in warm boot (also known as "wake from LP0") and cold boot (also known as "power up") sequences. Suggested actions for the Boot ROM/Boot Loader are noted after "Boot requirements:".

USAGE NOTE: The EMC register fields to be stored in the PMC must have "[PMC]", "[PMC2]", or "[PMC3]".in their comments. ([PMC2] registers are packed/unpacked after [PMC] register group in software code. [PMC3] is even later.) The fields that need to be stored in PMC Secure Scratch registers must have [PMC_SECURE] in their comments.

USAGE NOTE: When writing to a register, any bits that are not intended to be modified should be rewritten with their existing values.

USAGE NOTE: Unspecified bits may not appear in tables and should be written with their Reset values.

18.11.2.1 EMC_INTSTATUS_0

Interrupt Status Register

Clear on 1-write. Init value is clear. This register is “sticky”.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000xxx)

Bit	Reset	Description
9	CLEAR	DLL_LOCK_TIMEOUT_INT: Indicates a DLL lock timeout has occurred. In dual channel COMBINED_INTERRUPT_MODE, this is the OR of the two channels interrupt. 0 = CLEAR 1 = SET
8	CLEAR	CCFIFO_OVERFLOW_INT: Clock change FIFO overflow. In dual channel COMBINED_INTERRUPT_MODE, this is the OR of the two channels interrupt. 0 = CLEAR 1 = SET
7	CLEAR	DLL_ALARM_INT: Indicates a DLL alarm has been set. In dual channel COMBINED_INTERRUPT_MODE, this is the OR of the two channels interrupt. 0 = CLEAR 1 = SET
6	CLEAR	ACCESS_TO_SR_DPD_DEV_INT: Indicates a system attempt to access a self-refresh/deep-powered-down device. In dual channel COMBINED_INTERRUPT_MODE, this is the OR of the two channels interrupt. 0 = CLEAR 1 = SET
5	CLEAR	MRR_DIVLD_INT: MRR data is available to be read. In dual channel COMBINED_INTERRUPT_MODE, this is the OR of the two channels interrupt. 0 = CLEAR 1 = SET

Bit	Reset	Description
4	CLEAR	CLKCHANGE_COMPLETE_INT: CAR/EMC clock-change handshake complete. In dual channel COMBINED_INTERRUPT_MODE, this is the AND of the two channels interrupt. 0 = CLEAR 1 = SET
3	CLEAR	REFRESH_OVERFLOW_INT: Refresh request overflow timeout. In dual channel COMBINED_INTERRUPT_MODE, this is the OR of the two channels interrupt. 0 = CLEAR 1 = SET

18.11.2.2 EMC_INTMASK_0

Interrupt Mask Register

Init value is masked.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000xxx)

Bit	Reset	Description
9	MASKED	DLL_LOCK_TIMEOUT_INTMASK: Mask for DLL lock timeout interrupt. 0 = MASKED 1 = UNMASKED
8	MASKED	CCFIFO_OVERFLOW_INTMASK: Mask for clock change FIFO overflow. 0 = MASKED 1 = UNMASKED
7	MASKED	DLL_ALARM_INTMASK: Mask for DLL alarm and DLL alarm minimum. 0 = MASKED 1 = UNMASKED
6	MASKED	ACCESS_TO_SR_DPD_DEV_INTMASK: Mask for access a self-refresh/deep-powered-down device interrupt. 0 = MASKED 1 = UNMASKED
5	MASKED	MRR_DIVLD_INTMASK: Mask for MRR data available. 0 = MASKED 1 = UNMASKED
4	MASKED	CLKCHANGE_COMPLETE_INTMASK: Mask for CAR/EMC clock-change handshake complete. 0 = MASKED 1 = UNMASKED
3	MASKED	REFRESH_OVERFLOW_INTMASK: Mask for refresh request overflow timeout. 0 = MASKED 1 = UNMASKED

18.11.2.3 EMC_DBG_0

Debug Register

The DBG register is used to reconfigure the EMC during debug or chip testing.

Boot requirements: This register should be parameterized in the BCT and written by the Boot ROM during coldboot.

Boot requirements: This register should be saved in the scratch registers and restored by the Boot ROM during warmboot.

Offset: 0x8 I Read/Write: R/W I Reset: 0x01000c00 (0b00000001xxxxxxxxxx00110xxxx00000)

Bit	Reset	Description
31	DISABLED	AUTOCAL_IGNORE_SWAP: When set to ENABLED, autocal registers controlled by the autocal FSM do not obey the CFG_SWAP setting, and forces them to operate in the ACTIVE_ONLY mode. Setting to DISABLED will allow the autocal registers to obey the CFG_SWAP setting 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
30	DISABLED	WRITE_ACTIVE_ONLY: By default, when WRITE_MUX=ACTIVE, it writes both the shadowed and active version of the register. When WRITE_ACTIVE_ONLY is set, it only writes the ACTIVE copy. This overrides the programming of WRITE_MUX 0 = DISABLED 1 = ENABLED
29	DISABLED	ALLOW_HOSTIF_DURING_CCFIFO: Normally hostif requests are blocked when the CCFIFO is processing. Setting this to ENABLED unblocks them for debugging training/DVFS hangs 0 = DISABLED 1 = ENABLED
28	DISABLED	ALLOW_EMC1_PMACRO_CFG_ACCESS: When enabled, allows config access to EMC1 to be sent to the pad macro. By default, the pad macro configuration space is only visible through EMC0 0 = DISABLED 1 = ENABLED
27:26	ACTIVE_ONLY	CFG_SWAP: Enable the configuration swap feature. It is only intended to be used for the training sequence ACTIVE_ONLY means that the ASSEMBLY copy is copied into the ACTIVE copy SWAP means that the ASSEMBLY and ACTIVE copies swap values ASSEMBLY_ONLY means that the ACTIVE copy is copied into the ASSEMBLY copy (opposite of DISABLED) 0 = ACTIVE_ONLY 1 = SWAP 2 = ASSEMBLY_ONLY
25	DISABLED	AUTOCAL_ALLOW_WRITE_MUX: When DISABLED, autocal registers controlled by autocal FSM do not obey WRITE_MUX/WRITE_ACTIVE_ONLY settings, and operate in WRITE_MUX=ASSEMBLY/WRITE_ACTIVE_ONLY=DISABLED mode. Setting to ENABLED will allow autocal registers to obey WRITE_MUX/WRITE_ACTIVE_ONLY settings 0 = DISABLED 1 = ENABLED
24	ENABLED	CFG_PRIORITY: determines the priority of cfg accesses to the DRAM. Setting this register to ENABLED gives DRAM config cycles (refresh, mrs, emrs, etc.) higher priority over real time requestors. The DISABLED setting gives the real time requestors higher priority than DRAM config cycles. Do not program to DISABLED unless for debugging. 0 = DISABLED 1 = ENABLED
13	DISABLED	SUPPRESS_WRITE_CMD: suppress write command sent to DRAM 0 = DISABLED 1 = ENABLED
12	DISABLED	SUPPRESS_READ_CMD: suppress read command sent to DRAM 0 = DISABLED 1 = ENABLED
11	ENABLED	STRETCH_MRW_RESET:stretch MRW RESET command to three clocks; affects only LPDDR3 0 = DISABLED 1 = ENABLED
10	ENABLED	AP_REQ_BUSY_CTRL: determines whether the busy signal from the auto-precharge cancellation (APC) fifo is allowed to stall requests to the EMC. This field should usually be set to ENABLED. 0 = DISABLED 1 = ENABLED
9	MANAGED	READ_DQM_CTRL: controls whether the dqm signals during reads are managed for power (not relevant for DDR). If set to MANAGED, EMC only turns them on when necessary. If set to ALWAYS_ON, the dqm signals are enabled during non-write operation. This field should usually be set to MANAGED. 0 = MANAGED 1 = ALWAYS_ON
2	DISABLED	FORCE_UPDATE: causes the active state to get updated with the assembly state immediately upon setting this register to 1. 0 = DISABLED 1 = ENABLED
1	ASSEMBLY	WRITE_MUX: controls whether writes to the configuration registers are done from the assembly or active state. 0 = ASSEMBLY 1 = ACTIVE
0	ACTIVE	READ_MUX: controls whether reads to the configuration registers are done from the assembly or active state. 0 = ACTIVE 1 = ASSEMBLY

18.11.2.4 EMC_CFG_0

Configuration Register

The CFG register is used to configure the external memory interface.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xc | Read/Write: R/W | Reset: 0x03c00000 (0b0000x011110xx000xxxxxx000000000x)

Bit	Reset	Description
31	DISABLED	DRAM_CLKSTOP_PD: [PMC] clock stop is only allowed to happen if the DRAM is in active/precharge powerdown (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED
30	DISABLED	DRAM_CLKSTOP_SR: [PMC] clockstop is only allowed to happen if the DRAM is in self-refresh (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED
29	NO_POWERDOWN	DRAM_ACPD: [PMC] Allows the DRAM controller to perform opportunistic active powerdown control using the CKE pin on the DRAM. The behavior of the powerdown control logic is controlled by the PDEX2* and *2PDEN registers. The value of DRAM_ACPD should only be changed when CKE is low, e.g., during software-controlled self-refresh or before DRAM initialization. 0 = NO_POWERDOWN 1 = ACTIVE_POWERDOWN
28	DISABLED	DYN_SELF_REF: [PMC] 0 = DISABLED 1 = ENABLED
26	DISABLED	REQACT_ASYNC: [PMC] Allows independent transfers of activates and transactions from the MC to the EMC. Disabled to preserve the order of requests selected by mc_arb logic. 0 = DISABLED 1 = ENABLED
25	ENABLED	AUTO_PRE_WR: [PMC] enable using MC auto-precharge indication for writes. This bits, when set to DISABLE, will override the settings in the MC register (EMC ignore the auto-precharge indication from MC). Otherwise, they permit clients to make either auto or explicit precharge requests depend on the state of MAN_PRE_WR. 0 = DISABLED 1 = ENABLED
24	ENABLED	AUTO_PRE_RD: [PMC] enable using MC auto-precharge indication for reads. This bits, when set to DISABLE, will override the settings in the MC register (EMC ignores the auto-precharge indication from MC). Otherwise, they permit clients to make either auto or explicit precharge requests depend on the state of MAN_PRE_RD. 0 = DISABLED 1 = ENABLED
23	ENABLED	MAN_PRE_WR: [PMC] enable explicit-precharge in the EMC for writes. When this bit is enabled (and AUTO_PRE_WR=enable), EMC will issue an explicit precharge command after the memory write command. If this bit is disabled (and AUTO_PRE_WR=enable), the EMC will issue an auto-precharge with the memory write command. 0 = DISABLED 1 = ENABLED
22	ENABLED	MAN_PRE_RD: [PMC] enable explicit-precharge in the EMC for reads. When this bit is enabled (and AUTO_PRE_RD=enable), EMC will issue an explicit precharge command after the memory read command. If this bit is disabled (and AUTO_PRE_RD=enable), the EMC will issue an auto-precharge with the memory read command. 0 = DISABLED 1 = ENABLED
21	DISABLED	PERIODIC_QRST: [PMC] 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
18	DISABLED	DSR_VTTGEN_DRV_EN: [PMC] enable VTTGEN controls (via DSR_VTTGEN_DRV/ TXDSRVTTGEN registers) during DSR. 0 = DISABLED 1 = ENABLED
17:16	TWOX	EMC2MC_CLK_RATIO: [PMC] EMC to MC clock ratio. 0 = TWOX 1 = ONEX 2 = FOURX 3 = RESERVED
9	0x0	WAIT_FOR_ISP2B_READY_B4_CC: [PMC] 1 = wait for isp2b ready event to be asserted before acknowledge clock change.
8	0x0	WAIT_FOR_VI2_READY_B4_CC: [PMC] 1 = wait for vi2 ready event to be asserted before acknowledge clock change.
7	0x0	WAIT_FOR_ISP2_READY_B4_CC: [PMC] 1 = wait for isp2 ready event to be asserted before acknowledge clock change.
6	0x0	INVERT_DQM: [PMC] 1 = invert DQM polarity.
5	0x0	WAIT_FOR_DISPLAYB_READY_B4_CC: [PMC] 1 = wait for displayb ready event to be asserted before acknowledging clock change.
4	0x0	WAIT_FOR_DISPLAY_READY_B4_CC: [PMC] 1 = wait for display ready event to be asserted before acknowledging clock change.
3	0x0	EMC2PMACRO_CFG_BYPASS_DATAPIPE2: Reserved
2	0x0	EMC2PMACRO_CFG_BYPASS_DATAPIPE1: Reserved
1	0x0	EMC2PMACRO_CFG_BYPASS_ADDRPIPE: Reserved

18.11.2.5 EMC_ADR_CFG_0

External Memory Address Configuration System

The ADR_CFG register is used to specify the number of DRAM devices. DRAM density and geometry parameters are specified by the EMEM_ADR_CFG* registers in MC.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	N2	EMEM_NUMDEV: [PMC SECURE] Number of populated DRAM devices. 0 = N1 1 = N2

18.11.2.6 EMC_REFCTRL_0

Refresh Control Register

The REFCTRL register allows software to enable or disable the refresh controller.

REF_VALID should be enabled after the initialization sequence is completed.

Boot requirements:

- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized in the BCT and written by the Boot ROM during cold boot.
- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized the scratch registers and restored by the Boot ROM during warm boot.
- REF_VALID field should always be set to ENABLED for normal use, no need to parameterize in BCT/scratch.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
31	DISABLED	REF_VALID: Enable refresh controller. 0 = DISABLED 1 = ENABLED
1:0	0x0	DEVICE_REFRESH_DISABLE: Disables refresh to individual attached device (1 bit per DRAM chip select).

18.11.2.7 EMC_PIN_0

Controls State of Selected DRAM Pins

The PIN register allows software to control the state of the selected external DRAM pins.

Boot requirements:

- Both fields in this register should be set to NORMAL during cold boot.
- Both fields in this register should be set to NORMAL during warm boot.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xx00xxx0xxx0x000)

Bit	Reset	Description
17:16	DISABLED	PIN_GPIO_EN: General-purpose DRAM I/O enable 0 = DISABLED 1 = ENABLED
13:12	INACTIVE	PIN_GPIO: General-purpose DRAM I/O, to use corresponding EN must be set routing controlled via CMD_MAPPING_CMD* registers 0 = INACTIVE 1 = ACTIVE
8	ACTIVE	PIN_RESET: Selects the level of the RESET# pin (if applicable). 0 = ACTIVE 1 = INACTIVE
4	NORMAL	PIN_DQM: Is used to always mask DRAM writes. This pin should only be used for initialization. Certain DRAM vendors require the DQM to be high during initialization. The register value should be set to NORMAL after the initialization sequence. 0 = NORMAL 1 = INACTIVE
2	DISABLED	PIN_CKE_PER_DEV: per device CKE enable. When disabled, PIN_CKE is used for all CKE pins; when enabled PIN_CKEB is used for CKEB pins 0 = DISABLED 1 = ENABLED
1	POWERDOWN	PIN_CKEB: selects the level of the CKEB pin. This can be used to place the DRAM in power down state. PIN_CKEB value is applied to the CKEB pins when PIN_CKE_PER_DEV is 1. 0 = POWERDOWN 1 = NORMAL
0	POWERDOWN	PIN_CKE: Selects the level of the CKE pin. This can be used to place the DRAM in power down state. PIN_CKE value is applied to all CKE pins when PIN_CKE_PER_DEV is 0. 0 = POWERDOWN 1 = NORMAL

18.11.2.8 EMC_TIMING_CONTROL_0

Triggers an Update of the Timing-Related Registers

The TIMING_CONTROL register is used by software to trigger parameter updates for timing parameter registers, RDQS/QUSE delay controls, and some DLL controls. Writing the TIMING_UPDATE field updates the active state of these registers with the programmed assembly state. The active state is updated during a safe interval determined by the EMC. If CLKCHANGE_REQ_ENABLE is enabled, the active value will automatically be updated on completion of the clock change.

Note: *Programming of this register does not trigger the shadow register update event immediately. To prevent shadow register programming issued after programming this register from being*

latched accidentally, always poll for TIMING_UPDATE_STALLED==0 after programming this register.

Boot requirements:

- Writing this register with 0x1 will trigger a timing update event, which should be used in both warm boot and cold boot sequences.

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TIMING_UPDATE

18.11.2.9 EMC_RC_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2c | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
6:0	0xff	RC: [PMC] Specifies the row cycle time. This is the minimum number of cycles between activate commands to the same bank.

18.11.2.10 EMC_RFC_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x30 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxx0000111111)

Bit	Reset	Description
9:0	0x3f	RFC: [PMC] specifies the auto refresh cycle time. This is the minimum number of cycles between an auto refresh command and a subsequent auto refresh or activate command.

18.11.2.11 EMC_RAS_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x34 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
5:0	0x7f	RAS: [PMC] specifies the row active time. This is the minimum number of cycles between an activate command and a precharge command to the same bank.

18.11.2.12 EMC_RP_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x38 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RP: [PMC] specifies the row precharge time. This is the minimum number of cycles between a precharge command and an activate command to the same bank.

18.11.2.13 EMC_R2W_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3c | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	R2W: [PMC] specifies the minimum number of cycles from any read command to any write command, irrespective of bank. This parameter guarantees the read->write turnaround time on the bus.

18.11.2.14 EMC_W2R_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x40 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	W2R: [PMC] specifies the minimum number of cycles from a write command to a read command, irrespective of bank.

18.11.2.15 EMC_R2P_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	R2P: [PMC] specifies the minimum number of cycles from a read command to a precharge command for the same bank.

18.11.2.16 EMC_W2P_0

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. Counting starts from the last data transfer as related to where CAS_ would be if BL = 4.

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x48 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	W2P: [PMC] Specifies the minimum number of cycles from a write command to a precharge command for the same bank.

18.11.2.17 EMC_RD_RCD_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x4c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	RD_RCD: [PMC] Specifies the RAS to CAS delay. RD_RCD is the minimum number of cycles between an activate command and a read command to the same bank.

18.11.2.18 EMC_WR_RCD_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x50 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	WR_RCD: [PMC] Minimum number of cycles between an activate command and a write command to the same bank.

18.11.2.19 EMC_RRD_0

DRAM timing parameter

This register is shadowed: see usage note in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x54 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	RRD: [PMC] specifies the Bank X Act to Bank Y Act command delay.

18.11.2.20 EMC_REXT_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx00001)

Bit	Reset	Description
4:0	0x1	REXT: [PMC] specifies the read to read delay for reads when multiple physical devices are present.

18.11.2.21 EMC_WDV_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	WDV: [PMC] The number of cycles to post (delay) write data from being asserted to the RAMs. 40 = MAX

18.11.2.22 EMC_QUSE_0

DRAM timing parameter

Because SDRAM uses a tristating clock (the DQS), a method is needed to deal with the ambiguity of when DQS is tristated. Only one such method is supported: QUSE.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000010)

Bit	Reset	Description
5:0	0x2	QUSE: [PMC] Tells the chip when to look for read return data.

18.11.2.23 EMC_QRST_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxx00000xxxxxxxxx0000001)

Bit	Reset	Description
20:16	0x0	QRST_DURATION: [PMC] QRST remains asserted for QRST_DURATION + 1 cycles
6:0	0x1	QRST: [PMC] time from expiration of QSAFE until reset is issued.

18.11.2.24 EMC_QSAFE_0

DRAM timing parameter

When PERIODIC_QRST is enabled, the QSAFE parameter is intended to guarantee that the QRSTs will not interfere with pending reads (for example, the queue is empty). This field must be set to at least (RDV - QRST).

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000007 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000111)

Bit	Reset	Description
6:0	0x7	QSAFE: [PMC] Time from a read command to when it is safe to issue a QRST (delayed by the QRST parameter).

18.11.2.25 EMC_RDV_0

DRAM timing parameter

RDV is the read latency register. This register value is not negotiable; it will work with the right value and will not with the wrong value. It uses sequential timing, not combinational timing.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6c | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001000)

Bit	Reset	Description
6:0	0x8	RDV: [PMC] Time from read command to latching the read data from the pad macros. 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.26 EMC_REFRESH_0

DRAM timing parameter

The value of REFRESH is calculated using the following formula:

```
{REFRESH, REFRESH_LO} = max[(tREF/#_of_rows) / (emc_clk_period) - 64,
                            (tREF/#_of_rows) / (emc_clk_period) * 97%]
```

For example, if the clock frequency is 133 MHz, and the refresh requirement is 64 ms per 4096 rows. The programming value is 0x7e3.

This should always be programmed using the all-bank refresh timings, and does not need to change for per-bank refresh operation.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x70 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxx0000000000011111)

Bit	Reset	Description
15:6	0x0	REFRESH: [PMC] Specifies the interval between refresh requests.
5:0	0x1f	REFRESH_LO: [PMC]

18.11.2.27 EMC_BURST_REFRESH_NUM_0

DRAM timing parameter

BURST_REFRESH_NUM is used to specify the refresh burst count. The refresh controller will wait until BURST_REFRESH_NUM refreshes have been scheduled, then issue them all at once. This can result in a performance improvement in many cases.

Notes:

- *If tRAS(max) is less than the refresh interval (tREF/#_of_rows), tRAS(max) must be used instead of the refresh interval in the formula above. This is because refresh is used to satisfy tRAS(max) timing. Accordingly, BURST_REFRESH_NUM must be programmed in such a way that queuing up multiple refreshes does not violate tRAS(max) timing. Burst length = 2^BURST_REFRESH_NUM. Refreshes will be throttled to meet TREFBW limitation (8/window) if TREFBW > 0.*
- *Do not program this register to value non-zero unless for testing.*

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	BR1	BURST_REFRESH_NUM: [PMC] Specify the refresh burst count. 0 = BR1 1 = BR2 2 = BR4 3 = BR8 4 = BR16 5 = BR32 6 = BR64 7 = BR128 8 = BR256 9 = BR512 9 = MAX

18.11.2.28 EMC_PDEX2WR_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x78 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	PDEX2WR: Specify the timing delay from exit of powerdown mode to a write command.

18.11.2.29 EMC_PDEX2RD_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x7c | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	PDEX2RD: Specify the timing delay from exit of powerdown mode to a read command.

18.11.2.30 EMC_PCHG2PDEN_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x80 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	PCHG2PDEN: [PMC] Specify the timing delay from a precharge command to powerdown entry.

18.11.2.31 EMC_ACT2PDEN_0

[PMC] DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x84 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	ACT2PDEN: Specify the timing delay from an activate, MRS or EMRS command to power-down entry.

18.11.2.32 EMC_AR2PDEN_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x88 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxx000011111)

Bit	Reset	Description
8:0	0x1f	AR2PDEN: [PMC] Specify the timing delay from an autorefresh command to powerdown entry.

18.11.2.33 EMC_RW2PDEN_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x8c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0011111)

Bit	Reset	Description
6:0	0x1f	RW2PDEN: [PMC] Specify the timing delay from a read/write command to powerdown entry. Auto-precharge timing must be taken into account when programming this field

18.11.2.34 EMC_TXSR_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x90 | Read/Write: R/W | Reset: 0x000003fe (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111110)

Bit	Reset	Description
9:0	0x3fe	TXSR: [PMC] Cycles between self-refresh exit & first DRAM command that does not require a locked DLL. Largest allowed value is 0x3fe.

18.11.2.35 EMC_TCKE_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x94 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TCKE: Specify the minimum CKE high pulse width.

18.11.2.36 EMC_TFAW_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	TFAW: [PMC] specify the width of the FAW (four-activate window) for 8-bank devices. Set to 0 to disable this timing check. Only 4 activates may occur within the rolling window.

18.11.2.37 EMC_TRPAB_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	TRPAB: [PMC] Specify precharge-all tRP allowance for 8-bank devices. Setting this field to 0 will cause EMC to use TRP.TRP for precharge-all.

18.11.2.38 EMC_TCLKSTABLE_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001111)

Bit	Reset	Description
6:0	0xf	TCLKSTABLE: [PMC] Specify minimum number of cycles of a stable clock period prior to exiting powerdown or self-refresh modes.

18.11.2.39 EMC_TCLKSTOP_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa4 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4:0	0xf	TCLKSTOP: [PMC] Delay from last command to stopping the external clock to DRAM devices.

18.11.2.40 EMC_TREFBW_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	TREFBW: [PMC] specify the width of the burst-refresh window. If set to a non-zero value, only 8 refreshes will occur in this rolling window. Set to 0 to disable this timing check.

18.11.2.41 EMC_TPPD_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	TPPD: [PMC] precharge-precharge delay

18.11.2.42 EMC_ODT_WRITE_0

ODT may be enabled during writes via control of ODT pin.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxx0000xx000000)

Bit	Reset	Description
31	0x0	ENABLE_ODT_DURING_WRITE: enables ODT to be turned on prior to issuing write to DRAM. [PMC] If ENABLE_ODT_DURING_WRITE = 1 and DISABLE_ODT_DURING_READ = 0, ODT will always be enabled after the first write.
11:8	0x0	ODT_WR_DURATION: [PMC] for LPDDR3, indicate how long to assert ODT by.
5	0x0	SHARE_ONE_ODT: [PMC] For LPDDR3 only. When enabled, ODT[1] is not used. The ODT for both ranks will be sharing the same ODT[0]. When disabled, ODT[0] is controlling rank0, ODT[1] is controlling rank1.
4	0x0	DRIVE_BOTH_ODT: [PMC] If this field = 0, only the ODT for the requested device will be asserted during write. If this field = 1, ODTs for both devices will be asserted during write.
3:0	0x0	ODT_WR_DELAY: [PMC] Set this field = ABS (WL - ceiling(tAOND) - 2).

18.11.2.43 EMC_PDEX2MRR_0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb4 | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	PDEX2MRR: [PMC] precharge-precharge delay

18.11.2.44 EMC_WEXT_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the
- Boot ROM during warm boot.

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	WEXT: [PMC] specifies the write to write delay for writes when multiple physical devices are present.

18.11.2.45 EMC_RFC_SLR_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8:0	0x0	RFC_SLR: [PMC] Specifies the stagger refresh cycle time between ranks. This is the minimum number of cycles between an auto refresh command to one rank and a subsequent auto refresh to another rank. A zero value indicates stagger refresh is disabled.

18.11.2.46 EMC_MRS_WAIT_CNT2_0

This register allows the delay between an MRS/EMRS/MRW/MRR command and the following DRAM command

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xc4 | Read/Write: R/W | Reset: 0x0208000f (0bxxxxx01000001000xxxxxx0000001111)

Bit	Reset	Description
25:16	0x208	MRS_EXT2_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS ext2 command.
9:0	0xf	MRS_EXT1_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS ext1 command.

18.11.2.47 EMC_MRS_WAIT_CNT_0

This register allows the delay between an MRS/EMRS/MRW/MRR command and the following DRAM command.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xc8 | Read/Write: R/W | Reset: 0x0208000f (0bxxxxx01000001000xxxxxx0000001111)

Bit	Reset	Software Default	Description
26:16	0x208	0x3ff	MRS_LONG_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS long command.
9:0	0xf	0x3ff	MRS_SHORT_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS short command.

18.11.2.48 EMC_MRS_0

Command Trigger: MRS

The MRS register allows software to issue an MRS command.

BA0, BA1 are used to address MRS or EMRS registers in DRAM. Although this register can also program EMRS, use the EMRS register so that the hardware registers can shadow what is in the DRAM.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	MRS_DEV_SELECTN: [PMC] Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_MRS_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRS_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	MRS_BA: [PMC] Set to 0x0 for MRS.
13:0	0x0	MRS_ADR: [PMC] Mode-register data to be written.

18.11.2.49 EMC_EMRS_0

Command trigger: EMRS

The EMRS register allows software to issue an EMRS command.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xd0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxx0xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS_DEV_SELECTN: [PMC] Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_EMRS_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS_ADR: [PMC] Mode-register data to be written.

18.11.2.50 EMC_REF_0

Command Trigger: Refresh

> ***Note:*** *The REF register allows software to issue refresh commands. This is done to ensure proper DRAM initialization.*

Boot requirements:

- This register triggers a refresh command. REF_CMD should be written with 0x1 during cold boot to trigger refresh commands during DRAM initialization.

Offset: 0xd4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxx00000000000xxxxxx00)

Bit	Reset	Description
31:30	0x0	REF_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
18:8	0x0	REF_NUM: perform (REF_NUM + 1) refresh cycles.
1	0x0	REF_NORMAL: 0 = execute first refresh immediately (this is use during DRAM initialization). 1 = execute refresh through normal refresh mechanism (this should be used in normal operation).
0	0x0	REF_CMD: causes the hardware to perform a REFRESH to all DRAM banks.

18.11.2.51 EMC_PRE_0

Command Trigger: Precharge-All

The PRE register allows software to issue a precharge all command. This command may be used to ensure proper DRAM initialization.

Boot requirements:

- This register triggers a precharge-all command. PRE_CMD should be written with 0x1 during cold boot to trigger refresh commands during DRAM initialization.
- If per-device DPD is used, the PRE_DEV_SELECTN field should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:30	0x0	PRE_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	0x0	PRE_CMD: causes the hardware to perform a PRECHARGE to all DRAM banks.

18.11.2.52 EMC_NOP_0

Command Trigger: NOP

The NOP register allows software to issue an explicit NOP command. This command may be used to ensure proper DRAM initialization.

Boot requirements:

- This register triggers a no-operation command. NOP_CMD should be written with 0x1 during cold boot to trigger no-op commands during DRAM initialization.

Offset: 0xdc | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
31:30	0x0	NOP_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	0x0	NOP_CMD: causes the hardware to perform a NOP to all DRAM banks.

18.11.2.53 EMC_SELF_REF_0

Command Trigger: SELF REFRESH

The SELF_REF register allows software to issue self-refresh commands.

LP4: CKE follows SELF_REF_CMD

Do not program this register when a clock change sequence is on-going. Check the CLKCHANGE_COMPLETE_INT value if not sure.

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
31:30	0x0	SREF_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1, 0x3 for neither device.
8	DISABLED	ACTIVE_SELF_REF: Determines the state of the CKE pin while SELF_REF_CMD=ENABLE. If ENABLED, CKE is held true. This bit should only be ENABLED for DRAM types that support active self-refresh, e.g. LPDDR4. 0 = DISABLED 1 = ENABLED
0	DISABLED	SELF_REF_CMD: causes the hardware to issue a SELF_REFRESH command. REF_NUM refreshes will be issued to both ranks regardless of SREF_DEV_SELECTN. While SELF_REF_CMD=ENABLED, the state of the CKE pin is determined by ACTIVE_SELF_REF. The SELF_REF_CMD=ENABLED state will override the PIN:CKE setting. The DRAM will ignore all accesses until CMD:DISABLED. 0 = DISABLED 1 = ENABLED

18.11.2.54 EMC_DPD_0

Command Trigger: Deep Power Down

The DPD register allows software to issue a deep power down command.

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:30	0x0	DPD_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	DISABLED	DPD_CMD: causes the hardware to issue the deep power down command (Burst Terminate with CKE low). While in DPD mode, the DRAM will not maintain data integrity. While CMD:ENABLED, the CKE pin is held deasserted. The CMD:ENABLED state will override the PIN:CKE setting. The DRAM will ignore all accesses until CMD:DISABLED. 0 = DISABLED 1 = ENABLED

18.11.2.55 EMC_MRW_0

Command Trigger: MRW

Mode Register Write: LPDDRx-only version of MRS/EMRS

Boot requirements:

- This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000

Bit	Reset	Description
31:30	0x0	MRW_DEV_SELECTN: [PMC] Active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW_MA: [PMC] Register address
7:0	0x0	MRW_OP: [PMC] Data to be written

18.11.2.56 EMC_MRR_0

Command Trigger: MRR

Mode Register Read: For DRAM types supporting MRR (LPDDRx)

Sequence

1. 1. Read MRR until EMC_STATUS.MRR_DIVLD=0 (ok to skip if it is sure there are no pending MRR reads)
2. Write this register with the desired addr (MA) and device (DEV_SELECTN), device needs to be either DEV0 or DEV1: writing to both is illegal
3. Poll EMC_STATUS.MRR_DIVLD=1, or if using interrupt, wait for MRR_DIVLD_INT
4. Read back MRR. The value is in MRR_DATA field.

> ***Note:*** *It is okay to issue new MRR requests while there are on-going requests. For example, to issue 3 MRR requests, follow these steps: 1,2,2,2,3,4,3,4,3,4. Data read back in step 4 is in the same order requested in step 2.*

To make sure the EMC is available for new MRR requests, poll for EMC_STATUS.MRR_FIFO_SPACE > 0 before step 2.

If using 2 x16 DRAM, MRR_DATA[15:8] can be used to store MRR from the second DRAM on the same CS (configured via CFG_3.MRR_BYTESEL_X16)

Offset: 0xec | Read/Write: R/W | Reset: 0x0000XXXX (0b00xx00xx00000000xxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31:30	RW	0x0	MRR_DEV_SELECTN: active-low chip-select, choose which device to send the command to. (enum for safety). 0 = ILLEGAL 1 = DEV1 2 = DEV0 3 = RESERVED
27	RW	0x0	USE_MRR_EXT_CNT: 0 = USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1 = USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	RW	SHORT	USE_MRR_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	RW	0x0	MRR_MA: register address
15:0	RO	X	MRR_DATA: data returned

18.11.2.57 EMC_CMDQ_0

Command Queue Depth Register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This arbitration configuration register should be parameterized in the BCT and written by the OS during cold boot and warm boot.

Offset: 0xf0 | Read/Write: R/W | Reset: 0x10004408 (0bxxx10000xxxxxxxxx100x100xxx01000)

Bit	Reset	Description
28:24	0x10	RW_WD_DEPTH
14:12	0x4	PRE_DEPTH
10:8	0x4	ACT_DEPTH
4:0	0x8	RW_DEPTH

18.11.2.58 EMC_MC2EMCQ_0

Command Queue Depth Register

Boot requirements:

- This arbitration configuration register should be parameterized in the BCT and written by the OS during cold boot and warm boot.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xf4 | Read/Write: R/W | Reset: 0x06000404 (0bxxxx0110xxxxxxxxxxxxx100xxxxx100)

Bit	Reset	Description
27:24	0x6	MCWD_DEPTH
10:8	0x4	MCACT_DEPTH
2:0	0x4	MCREQ_DEPTH

18.11.2.59 EMC_FBIO_SPARE_0

FBIO Spare Register

Note: *CFG_FBIO_SPARE_0[1] = FBIO_SPARE[1] = lock further change to ADDR_SWIZZLE_STACK* and ADR_CFG. registers.*

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- CFG_FBIO_SPARE_3 should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	CFG_FBIO_SPARE_3
23:16	0x0	CFG_FBIO_SPARE_2
15:8	0x0	CFG_FBIO_SPARE_1
7:2	0x0	CFG_FBIO_SPARE_0
1	0x0	CFG_ADR_EN: [PMC_SECURE] write once after powering up EMC.
0	0x0	CFG_SWAP_DLL: [PMC]

18.11.2.60 EMC_FBIO_CFG5_0

FBIO Configuration Register

The trimmer value may be overridden by setting using MULT==0 and OFFS = value << 4.

The FBIO_CFG5 register controls the FBIO I/O cells.

The following fields are shadowed: DIFFERENTIAL_DQS, CTT_TERMINATION, DQS_PULLD, CMD_2T_TIMING.

Writes to these fields will not take effect until the active value is updated via TIMING_UPDATE or (if enabled) CLKCHANGE_REQ.

Boot requirements:

- This register (except for field DISABLE_CONCURRENT_AUTOPRE) should be parameterized in the BCT and written by the Boot ROM during cold boot.

- This register (except for field DISABLE_CONCURRENT_AUTOPRE) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x104 | Read/Write: R/W | Reset: 0x10400015 (0b00x100000100xxxx0000x0x0xxx10101) | Default: 0x00400010

Bit	Reset	Software Default	Description
31	DISABLED	NONE	DATA_BUS_RETURN_TO_ZERO: [PMC] this will drive the data bus back to a zero as resting stage, which can save power in LP4. Shadowed 0 = DISABLED 1 = ENABLED
30	DISABLED	NONE	DATA_BUS_RETURN_TO_ONE: [PMC] this will drive the data bus back to a one as resting stage, which can save power in LP3. Shadowed 0 = DISABLED 1 = ENABLED
28	ENABLED	NONE	MASK_PUTERM_N_DQS_PULLD_DURING_ZQCAL: [PMC] When this bit is enabled; it deasserts DQ/DQS puterm and DQS pulld during ZQ calibration in hardware. 0 = DISABLED 1 = ENABLED
27	DISABLED	_NONE_	CMD_BUS_RETURN_TO_ZERO: [PMC] this will drive the cmd bus back to a zero as resting stage, which is needed as deselect during CKE transistions in LP4. Shadowed 0 = DISABLED 1 = ENABLED
26	DISABLED	NONE	CMD_BUS_RETURN_TO_ONE: [PMC] this bit drives the cmd bus back to a one as a resting stage (legacy Tegra device behavior). Power is saved if this is not done (Tegra X1 behavior). Shadowed 0 = DISABLED 1 = ENABLED
25	DISABLED	NONE	LPDDR3_DRAM: [PMC] Enables differentiation between lpddr3 DRAM protocol. 0 = DISABLED 1 = ENABLED
24	DISABLED	NONE	LPDDR3_WR_PREAMBLE_TOGGLE: [PMC] enable LPDDR3 write preamble toggle. 0 = DISABLED 1 = ENABLED
23:20	0x4	0x4	ERR_RD_BUBBLE: [PMC] Number of bubbles to be inserted in CMDQ after every error-Read. A dummy read is placed in CMDQ for each error read. This field controls the number of emcclk bubbles inserted for each dummy read. It is expected that this should be set to BL/2. Without this, ret_stat_fifo will build up and block non-error reads. This could cause ret_data_fifo to overflow as it is only 8 deep (area savings) whereas ret_stat_fifo is 20 deep. Shadowed. Not applicable if REQ_CTRL.KILL_DECERR_READS=0.
15:13	NORMAL	NONE	EMC2PMACRO_CFG_QUSE_MODE: [PMC] Select QUSE mode. Shadowed. (NORMAL = off-chip) (ALWAYS_ON = requires DQS PULL) (INTERNAL_LPBK selects on-chip loopback) (DIRECT_QUSE = directly controlled QUSE). 0 = NORMAL 1 = ALWAYS_ON 2 = INTERNAL_LPBK 3 = PULSE_INT 4 = RESERVED 5 = DIRECT_QUSE 6 = RESERVED1
12	DISABLED	NONE	CMD_2T_TIMING: [PMC] Enable 2T command timing (RAS/CAS/WE/ROW/A/BA). Shadowed. 0 = DISABLED 1 = ENABLED

Bit	Reset	Software Default	Description
10	DISABLED	NONE	DISABLE_CONCURRENT_AUTOPRE: [PMC] Disables reads/writes to a device until the precharge command has been issued by the DRAM internally. 0 = DISABLED 1 = ENABLED
8	ENABLED	ENABLED	CMD_TX_EN: [PMC] controls tx_en on the cmd bricks, active low 0 = ENABLED 1 = DISABLED
4	X64	X64	DRAM_WIDTH: [PMC] Specifies DRAM width. Platform specific. 0 = X32 1 = X64
3:2	BURST8	NONE	DRAM_BURST: [PMC] Specifies the burst length to use for the attached device(s). Shadowed. 0 = BURST4 1 = BURST8 2 = Reserved 3 = BURST16
1:0	LPDDR4	NONE	DRAM_TYPE: [PMC] Specifies which DRAM protocol to use for the attached device(s). Should be set to LPDDR2 for LPDDR3. 0 = DDR3 1 = LPDDR4 2 = LPDDR2 3 = DDR2

18.11.2.61 EMC_FBIO_CFG6_0

FBIO Configuration Register

QUSE_LATE determines how much added delay FBIO should add to the QUSE path. QUSE needs to align (approximately) with the incoming DQS in order to qualify it, since the incoming DQS/feedback clock is not always valid.

DRAMC can position QUSE with m2clk granularity (2 bit times). QUSE_LATE provides finer granularity of 1/2 an M2CLK cycle (1/2 bit time). The amount of delay added is primarily a function of the round trip wire delay to/from the DRAM. Other portions of the delay (driver and receiver delay) are compensated for by delay through a non-bonded QUSE pad cell.

Additional delay can be added to QUSE vi XFORM_QUSEx_MULT and XFORM_QUSEx_OFFS (applied to DLL offset).

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CFG_QUSE_LATE: [PMC]

18.11.2.62 EMC_PDEX2CKE_0

DRAM timing parameter

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	PDEX2CKE: [PMC] specifies the timing delay from exit of powerdown mode to turning on CKE

18.11.2.63 EMC_CKE2PDEN_0

DRAM timing parameter

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x11c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	CKE2PDEN: [PMC] specifies the timing delay from turning off CKE to powerdown entry.

18.11.2.64 EMC_CFG_RSV_0

EMC Configuration Reserved

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x120 | Read/Write: R/W | Reset: 0xff00ff00 (0b11111111000000001111111100000000)

Bit	Reset	Description
31:24	0xff	CFG_RESERVED_BYTE3
23:16	0x0	CFG_RESERVED_BYTE2
15:8	0xff	CFG_RESERVED_BYTE1
7:0	0x0	CFG_RESERVED_BYTE0

18.11.2.65 EMC_ACPD_CONTROL_0

Threshold for ACPD

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	ACPD_THRESHOLD: [PMC] number of idle cycles to wait before allowing power-down entry

18.11.2.66 EMC_MPC_0

MPC cmd is only supported for LP4.

Offset: 0x128 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
31:30	0x0	MPC_DEV_SELECTN
27:26	0x0	MPC_SUBP_SELECTN: Active-low subchannel select 0x0 applies command to both sub-channels 0x1 for subp1 0x2 to for only subp0
9	0x0	MPC_WR: Send CAS2 (required for FIFO WR)
8	0x0	MPC_RD: Send CAS2 (required for FIFO RD and DQ_CAL RD)

Bit	Reset	Description
7	0x0	MPC_CAS2: Send CAS2 (required for FIFO RD/WR and DQ_CAL RD)
6:0	0x0	MPC_OP: MPC OP code

18.11.2.67 EMC_EMRS2_0

Command Trigger: EMRS2

The EMRS2 register allows software to issue an EMRS2 command.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS2_DEV_SELECTN: [PMC] active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS2_EXT_CNT: [PMC]{USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_EMRS2_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS2_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS2_ADR: [PMC] mode-register data to be written.

18.11.2.68 EMC_EMRS3_0

Command Trigger: EMRS2

The EMRS3 register allows software to issue an EMRS3 command.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS3_DEV_SELECTN: [PMC] active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS3_EXT_CNT: [PMC]: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_EMRS3_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS3_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS3_ADR: [PMC] mode-register data to be written.

18.11.2.69 EMC_MRW2_0

Command Trigger: MRW2

Mode Register Write: LPDDRx -only version of MRS/EMRS

Boot requirements:

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW2_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW2_EXT_CNT: [PMC] 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW2_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW2_MA: [PMC] register address
7:0	0x0	MRW2_OP: [PMC] data to be written

18.11.2.70 EMC_MRW3_0

Command Trigger: MRW3

Mode Register Write: LPDDRx -only version of MRS/EMRS

Boot requirements:

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW3_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW3_EXT_CNT: [PMC] 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW3_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW3_MA: [PMC] register address
7:0	0x0	MRW3_OP: [PMC] data to be written

18.11.2.71 EMC_MRW4_0

Command Trigger: MRW4

Mode Register Write: LPDDRx -only version of MRS/EMRS

Boot requirements:

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW4_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW4_EXT_CNT: [PMC] 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW4_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW4_MA: [PMC] register address
7:0	0x0	MRW4_OP: [PMC] data to be written

18.11.2.72 EMC_CLKEN_OVERRIDE_0

Second-Level Clock Enable Override Register

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxx000xx000x)

Bit	Reset	Description
31	DISABLED	OBS_BUS_CLKEN: 0 = DISABLED 1 = ENABLED
8	CLK_GATED	PAD_CONFIG_OVR: [PMC] 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
7	CLK_GATED	TR_CLKEN_OVR: [PMC] 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
6	CLK_GATED	STATS_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
3	CLK_GATED	RR_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
2	CLK_GATED	DRAMC_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
1	CLK_GATED	CMDQ_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

18.11.2.73 EMC_R2R_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	R2R:

18.11.2.74 EMC_W2W_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	W2W:

18.11.2.75 EMC_EINPUT_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	EINPUT: [PMC] specifies when to assert EINPUT for a read, should normally be the same as QUSE.

18.11.2.76 EMC_EINPUT_DURATION_0

DRAM Timing Parameter

The minimum for EINPUT_DURATION is 4.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x150 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000100)

Bit	Reset	Description
5:0	0x4	EINPUT_DURATION: [PMC] Specifies how long the EINPUT should be asserted.

18.11.2.77 EMC_PUTERM_EXTRA_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000001)

Bit	Reset	Description
5:0	0x1	RXTERM: [PMC] tells the chip when to assert dynamic TERM for read return data.

18.11.2.78 EMC_TCKESR_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x158 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TCKESR: [PMC] Specify minimum low CKE pulse width for self-refresh mode.

18.11.2.79 EMC_TPD_0

DRAM Timing Parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x15c | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TPD: [PMC] specify minimum low CKE pulse width for power-down mode.

18.11.2.80 EMC_AUTO_CAL_CONFIG_0

Auto-Calibration Settings for EMC Pads

Boot requirements:

- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x0a195048 (0b00001010000110010101000001001000)

Bit	R/W	Reset	Description
31	RO	0x0	AUTO_CAL_START: [PMC] Writing a one to this bit triggers a complete Autocal cycle starting with measure FSM 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
30	RW	DISABLE	AUTO_CAL_COMP_PAD_FLIP: [PMC] When enabled, flip the COMP pad slices used for pull-up and pull-down calibration. Meant to address any full-chip connectivity/board routing issues related to off-chip calibration resistors 0 = DISABLE 1 = ENABLE
29	RW	DISABLED	AUTO_CAL_ENABLE: [PMC] ENABLED (normal operation): Use EMC Autocal-generated pullup/dn cal codes for drive/termination strength. DISABLED: Use emc padmacro register settings for pullup/dn drive/termination strength. When changed from ENABLED -> DISABLED, measure/compute/transfer FSMs immediately go to idle state. Old cal codes are still retained in padmacro registers and continue to be used till they are overwritten 0 = DISABLED 1 = ENABLED
28:25	RW	0x5	AUTO_CAL_NUM_SAMPLES: [PMC] If AUTO_CAL_CHECK_LOCK is enabled, number of samples to take for the majority test
24	RW	DISABLE	AUTO_CAL_CHECK_LOCK: [PMC] If enabled, whenever COMP pad indicates a "lock" take multiple samples and latch cal code only if majority of samples indicate "lock". Step to the next higher test value if majority of samples indicate "no lock" 0 = DISABLE 1 = ENABLE
23:19	RW	0x3	AUTO_CAL_WAIT_AFTER_EN: [PMC] Wait time between enabling COMP pad (i.e. changing TX_EN, BG_E_PWRD, IVREF_CAL_MODE, IVREF_LVL, E_IVREF or E_INPUT) and changing TX_DRVUP/TX_DRVDN test cal codes (in microseconds)
18:16	RW	0x1	AUTO_CAL_STEP: [PMC] Autocal measure step interval (in microseconds)
15:11	RW	0xa	AUTO_CAL_UPDATE_DELAY: [PMC] Number of EMC clocks to wait before Autocal transfer FSM triggers an update FSM cycle. This delay accounts for worst case priv write latency from EMC core to farthest pad macros. Static - Do not change or randomize
10	RW	DISABLE	AUTO_CAL_UPDATE_STALL: [PMC] When enabled, Autocal update FSM exits any ongoing update cycle and goes to idle state. Cal codes from previously completed update cycle are still available and continue to be used 0 = DISABLE 1 = ENABLE
9	RW	DISABLE	AUTO_CAL_MEASURE_STALL: [PMC] When enabled, Autocal measure FSM exits any ongoing vref cycle and goes to idle state. Locked cal codes from previously completed vref cycles are still available and continue to be used 0 = DISABLE 1 = ENABLE
8	RW	DISABLE	AUTO_CAL_PD_VREF2_EN: [PMC] Enable vref2 pull-down calibration measure cycle 0 = DISABLE 1 = ENABLE
7	RW	DISABLE	AUTO_CAL_PD_VREF1_EN: [PMC] Enable vref1 pull-down calibration measure cycle 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	AUTO_CAL_PD_VREF0_EN: [PMC] Enable vref0 pull-down calibration measure cycle 0 = DISABLE 1 = ENABLE
5	RW	DISABLE	AUTO_CAL_PU_VREF2_EN: [PMC] Enable vref2 pull-up calibration measure cycle 0 = DISABLE 1 = ENABLE
4	RW	DISABLE	AUTO_CAL_PU_VREF1_EN: [PMC] Enable vref1 pull-up calibration measure cycle 0 = DISABLE 1 = ENABLE
3	RW	ENABLE	AUTO_CAL_PU_VREF0_EN: [PMC] Enable vref0 pull-up calibration measure cycle 0 = DISABLE 1 = ENABLE
2	RO	0x0	AUTO_CAL_UPDATE_START: [PMC] Writing a one to this bit triggers an Autocal update FSM cycle 0 = DISABLE 1 = ENABLE
1	RO	0x0	AUTO_CAL_TRANSFER_START: [PMC] Writing a one to this bit triggers an Autocal transfer FSM cycle 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
0	RO	0x0	AUTO_CAL_COMPUTE_START: [PMC] Writing a one to this bit triggers an Autocal compute FSM cycle 0 = DISABLE 1 = ENABLE

18.11.2.81 EMC_AUTO_CAL_INTERVAL_0

EMC Pad Calibration Interval

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:0	0x0	AUTO_CAL_INTERVAL: [PMC] Auto-calibration occurs at intervals equivalent to the programmed number of microseconds.

18.11.2.82 EMC_AUTO_CAL_STATUS_0

Autocal master status register

Offset: 0x2ac | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:24	X	AUTO_CAL_VREF0_DRVDN: Pull-down code from VREF0 measure cycle
21:16	X	AUTO_CAL_VREF0_DRVUP: Pull-up code from VREF0 measure cycle
9	X	AUTO_CAL_PD_CAL_ZI: RX_D pin of COMP pad slice used for pull-down calibration
8	X	AUTO_CAL_PU_CAL_ZI: RX_D pin of COMP pad slice used for pull-up calibration
4	X	AUTO_CAL_ACTIVE: One when Autocal is active. Superset of measure + compute + transfer + update FSM cycles
3	X	AUTO_CAL_UPDATE_ACTIVE: One when Autocal update FSM cycle is active
2	X	AUTO_CAL_TRANSFER_ACTIVE: One when Autocal transfer FSM cycle is active
1	X	AUTO_CAL_COMPUTE_ACTIVE: One when Autocal compute FSM cycle is active
0	X	AUTO_CAL_MEASURE_ACTIVE: One when Autocal measure FSM cycle is active

18.11.2.83 EMC_REQ_CTRL_0

Request Status/Control

When either STALL_ALL_READS and/or STALL_ALL_WRITES is asserted, the stalling of read and/or write requests will not take effect until the status bit from EMC_STATUS register's NO_OUTSTANDING_TRANSACTIONS field is asserted.

Offset: 0x2b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx00)

Bit	Reset	Description
1	0x0	STALL_ALL_WRITES: Stall incoming write transactions
0	0x0	STALL_ALL_READS: Stall incoming read transactions

18.11.2.84 EMC_EMC_STATUS_0

EMC State-Machine Status

DRAM_IN_POWERDOWN, DRAM_IN_SELF_REFRESH, DRAM_IN_DPD: active high signal indicating current DRAM status for each of these modes, with 1 status bit per device, bit[0] = dev0 status, bit[1] = dev1 status.

Example: DRAM_IN_SELF_REFRESH = 0x3, both devices are in self-refresh, DRAM_IN_DPD= 0x2 would indicate only dev1 is in deep-power-down mode.

> ***Note:*** *If EMC is reset or powered down, the actual DRAM state could be different than indicated by these status bits. These bits do not reflect manually entered/exited powerdown or self-refresh (via use of PIN_CKE).*

Offset: 0x2b4 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:26	X	ACPD_FSM_IDLE: Indicate dev[n] power-down FSM is idle.
25:24	X	DSR_FSM_IDLE: Indicate dev[n] dynamic self refresh FSM is idle.
23	X	TIMING_UPDATE_STALLED: Indicates timing update triggered by TIMING_CONTROL.TIMING_UPDATE is not yet completed.
22	X	ZQ_FSM_IDLE: Indicates auto ZQ state machine is idle.
21	X	CFG_ZQ_ACTIVE: Indicates ZQ configuration access is active.
20	X	MRR_DIVLD: MRR data available for reading
19:16	X	MRR_FIFO_SPACE: MRR FIFO space available
13:12	X	DRAM_IN_DPD: dev[n] has been put into deep powerdown state
11:10	X	DRAM_IN_ACTIVE_SELF_REFRESH: dev[n] has been put into ACTIVE self-refresh (will remain until SR exit command).
9:8	X	DRAM_IN_SELF_REFRESH: dev[n] has been put into self-refresh (will remain until SR exit command). Indicates ACTIVE/POWERDOWN self-refresh
5:4	X	DRAM_IN_POWERDOWN: dev[n] has entered powerdown state (incoming req's will awaken if not stalled)
2	X	NO_OUTSTANDING_TRANSACTIONS: All non-stalled requests have completed
0	X	EMC_REQ_FIFO_EMPTY: Request FIFO is empty

18.11.2.85 EMC_CFG_2_0

EMC Configuration

Clock change sequencing: Once the divider is reprogrammed, CAR signals to EMC that a clock change is pending. If enabled, EMC stalls incoming requests, drains outstanding requests, and, if CLKCHANGE_(PD|SR)_ENABLE is enabled, puts DRAM into power-down (or self-refresh) before signalling to CAR that it is idle and ready for the change to happen. CAR will then change the divider/pll reprogramming. Once complete, EMC updates its shadow registers (assuming they may have been reprogrammed for new clock setting), unstalls requests, and resumes operation with new clock settings.

Some fields of this register are shadowed: see usage note in Section 18.11.2: EMC Registers.

Boot requirements:

- This register (except for fields DRAMC_PRE_B4_ACT, MRR_BYTESEL_X16, MRR_BYTESEL) should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x2b8 | Read/Write: R/W | Reset: 0x00108017 (0b0000000000x1xxx010xx000000010111)

Bit	Reset	Description
31	DISABLED	DRAMC_PRE_B4_ACT: [PMC] Diagnostic bit, gives priority to activates over precharges, determining which (precharge/activate) is processed first if both are pending and unblocked. 0 = DISABLED 1 = ENABLED
30	DISABLED	IGNORE_MC_A_BUS: [PMC] Tell whether to ignore the MC *_A_* bus. 0 = DISABLED 1 = ENABLED
29	DISABLED	CLR_ACT_BANK_INUSE_WHEN_BANK_CLOSE: [PMC] Tell whether to close/unlock actfifo bank inuse bit when the bank is close. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
28	DISABLED	DONT_CLR_TIMING_COUNTER_WHEN_CLKCHANGE: [PMC] Don't clear timing counters when clock change happen. 0 = DISABLED 1 = ENABLED
27:26	0x0	DRAMC_WD_CHK_POLICY: [PMC] 00 = no need to check if WDV (adj) >= WDV_CHK_BASE - Shadowed 01 = no need to check if WDV (adj) >= WDV_CHK_BASE+1 10 = no need to check if WDV (adj) >= WDV_CHK_BASE+2 11 = always check RWAQ_WD fifo quantity where WDV (adj) is in terms of EMC clocks
25	DISABLED	ALLOW_REF_DURING_CC_PRE_EXE: [PMC] Allow refresh to happen during clock change pre-execute phase. 0 = DISABLED 1 = ENABLED
24	DISABLED	DSR_STUTTER_ENABLE: [PMC] 1 = enable DSR stutter mode/protocol between MC and EMC. 0 = DISABLED 1 = ENABLED
23	0x0	CHK_PDEX2RD_TO_START_WR: [PMC] 1 = check PDEX2RD when starting write. 0 = check PDEX2WR when starting write. Shadowed
22	0x0	ISSUE_PCHGALL_AFTER_REF: [PMC] 1 = always generate precharge all bank after refresh. Shadowed
20	ENABLED	COMBINED_INTERRUPT_MODE: [PMC] 1 = combined interrupt mode. 0 = independent interrupt mode. Shadowed 0 = DISABLED 1 = ENABLED
16	DISABLED	CLKCHANGE_ACTIVE_SR: [PMC] execute frequency change when DRAM is in active SELF-REF state 0 = DISABLED 1 = ENABLED
11	0x0	DIS_CNTR_WITH_CFG_TIMING_UPDATE: [PMC] Diagnostic bit. 1 = Disable reset of timing parameter counters with TIMING_CONTROL.TIMING_UPDATE.
7	0x0	EARLY_TRFC_8_CLK: [PMC] Diagnostic bit. Shadowed. 1 = 8 clocks early 0 = 16 clocks early.
5:3	0x2	ZQ_EXTRA_DELAY: [PMC] Additional delay to push out ZQCMD based on tODToff (maximum) & frequency.
2	0x1	REF_AFTER_SREF: [PMC] 1 = Enables trigger of refresh after exiting self-refresh.
1	ENABLED	CLKCHANGE_PD_ENABLE: [PMC] Forces DRAM into power-down during CLKCHANGE. 0 = DISABLED 1 = ENABLED
0	ENABLED	CLKCHANGE_REQ_ENABLE: [PMC] Allows the EMC and CAR to handshake on PLL divider/source changes. 0 = DISABLED 1 = ENABLED

18.11.2.86 EMC_CFG_DIG_DLL_0

Configure Digital DLL

Controls for digital DLL's, which used to measure and maintain 1/4 cycle phase adjustment for RDQS strobes.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register (except for field CFG_DLL_ALARM_DISABLE and DLL_RESET (trigger) and CFG_DLL_USE_OVERRIDE_UNTIL_LOCK(trigger)) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for field CFG_DLL_ALARM_DISABLE and DLL_RESET (trigger) and CFG_DLL_USE_OVERRIDE_UNTIL_LOCK(trigger)) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2bc | Read/Write: R/W | Reset: 0xXX0000XX (0bx0xx0x00000000000000x000011x11x0)

Bit	R/W	Reset	Description
31	RO	X	CFG_DLL_USE_OVERRIDE_UNTIL_LOCK: [PMC] Writing 1 to this register causes override_val to be used in place of DLL output until DLL_LOCK/DLL_LOCK_TIMEOUT is obtained. Takes effect on next shadow update.
30	RO	0x0	DLL_RESET: [PMC] Writing 1 to this register will send reset pulse to DLL's on next shadow update. Must reset DLLs when changing clock frequency by factor >= 2. Reset occurs at next shadow update.
27	RW	0x0	CFG_DLL_ALARM_DISABLE: [PMC] Diagnostic bit. Shadowed. Disable override of DLL logic when DLL_ALM is set (otherwise overrides DLI to 0x3FF).
26	RO	X	CFG_DLL_UPDATE_AT_NXT_REFRESH: Writing 1 to this register causes the DLCELL update to occur at the next REFRESH interval. Ordinarily, updates are only performed if the DLL value is updated. This mechanism allows updates to the XFORM_MULT/OFF to be taken at the next REFRESH.
25:16	RW	0x0	CFG_DLL_OVERRIDE_VAL: [PMC] Value to use in place of DLI output if CFG_DLL_OVERRIDE_EN is set or prior to lock in RUN_TIL_LOCK mode. Shadowed.
15	RW	0x0	CFG_DLL_TESTEN: [PMC] Enable DLL test mode
14	RW	0x0	CFG_DLL_QUSE_TESTOUT: [PMC] Enable DLL TESTOUT on QUSE pads
13:12	RW	0x0	CFG_DLL_TESTSEL: [PMC] Select DLL test output
10:8	RW	0x0	CFG_DLL_UDSET: [PMC] In case DLL has problems locking. Shadowed. DLL will be treated as locked after LIMIT uSec. Counter is reset with DLL_RESET (from above) or with each periodic update (if using RUN_PERIODIC). Settings are: 000: LIMIT = 16 µs 001: LIMIT = 64 µs 010: LIMIT = 12 µs 011: LIMIT = 512 µs 011: LIMIT = 1
7:6	RW	RUN_TIL_LOCK	CFG_DLL_MODE: [PMC] Controls how frequently DLL runs. Shadowed. 0 = RUN_CONTINUOUS: DLL will run continuously. This option will consume the most power. Once LOCK/TIMEOUT occurs, trimmers will be updated at REFRESH or shadow update. 1 = RUN_TIL_LOCK: after DLL_RESET is set, DLL will run until it has locked/timed out, then be disabled. Trimmers updated at following REFRESH/shadow update. 2 = RUN_PERIODIC: after DLL_LOCK/TIMEOUT, DLL will be disabled and re-enabled after CFG_DLL_RUN_PERIOD µs to track delay changes (temperature/voltage). 3 = RUN_PRELOCK
5	RW	0x1	CFG_DLL_LOWSPEED: [PMC] Enable DLL for use with low-speed EMCCLK operation (< 200 MHz). Shadowed. The DLL does power optimization by itself. This control should stay at 1.
4	RO	X	CFG_DLL_STALL_RW_UNTIL_LOCK: [PMC] Writing 1 to this register will cause the EMC to stall all RW traffic until the DLL locks. Takes effect on the next shadow update.
3	RW	0x1	CFG_DLL_STALL_ALL_TRAFFIC: [PMC] Enables stalling of all DRAM traffic while undergoing dll_stall condition;otherwise only block reads/writes are stalled during dll_stall. Shadowed.
2	RW	ENABLED	CFG_DLL_OVERRIDE_EN: [PMC] Override DLL's DLI output with OVERRIDE_VAL (still uses mult/offset). Shadowed. 0 = DISABLED 1 = ENABLED
1	RO	X	CFG_DLL_STALL_ALL_UNTIL_LOCK: [PMC] Writing 1 to this register will cause emc to stall all traffic until DLL locks. Takes effect on next shadow update.
0	RW	DISABLED	CFG_DLL_EN: [PMC] Enables digital DLL. Shadowed. 0 = DISABLED 1 = ENABLED

18.11.2.87 EMC_CFG_DIG_DLL_PERIOD_0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	CFG_DLL_RUN_PERIOD: [PMC] If CFG_DLL_MODE == RUN_PERIODIC, this specifies the interval between runs in microseconds.

18.11.2.88 EMC_DIG_DLL_STATUS_0

Digital DLL Status

Digital DLL Status bits directly from DLL

Offset: 0x2c4 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	DLL_ALARM_MIN
17	X	DLL_PRIV_UPDATED
16	X	DLLCAL_IN_PROGRESS
15	X	DLL_LOCK
14	X	DLL_ALARM
13	X	DLL_LOCK_TIMEOUT
10:0	X	DLL_OUT

18.11.2.89 EMC_CFG_DIG_DLL_1_0

Digital DLL Status

Additional DLL configuration bits used by DRAMc update logic

Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00003801 (0bxxxxxxxxxxxxxxxx0011100000000001)

Bit	Reset	Description
15:12	0x3	CFG_DLL_WAIT_BEFORE_UPDATE: [PMC] Number of clocks to stall before DLL update sent to pad macros. Minimum valid value is 0x3. Shadowed
11:6	0x20	DLL_NUM_STALL_CYCLES: [PMC] Number of clocks to stall after DLL update sent to pad macros. Shadowed
5:1	0x0	CFG_DLL_UPDATE_TIMEOUT: [PMC] Timeout if waiting for idle/refresh- 0 means no timeout. Shadowed
0	0x1	CFG_DLL_UPDATE_IDLE: [PMC] Update Dig Dll during idle (outside refresh). Shadowed

18.11.2.90 EMC_RDV_MASK_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2cc | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0111111)

Bit	Reset	Description
6:0	0x3f	RDV_MASK: [PMC] This should be programmed to RDV.

18.11.2.91 EMC_WDV_MASK_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001111)

Bit	Reset	Description
5:0	0xf	WDV_MASK: [PMC] This should be programmed to WDV.

18.11.2.92 EMC_RDV_EARLY_MASK_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2d4 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0111111)

Bit	Reset	Description
6:0	0x3f	RDV_EARLY_MASK: [PMC] This should be programmed to RDV_EARLY.

18.11.2.93 EMC_RDV_EARLY_0

DRAM timing parameter

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0001000)

Bit	Reset	Description
6:0	0x8	RDV_EARLY: [PMC] time from read command to latching the read data from the pad macros. 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.94 EMC_AUTO_CAL_CONFIG8_0

Autocal master compute control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2dc | Read/Write: R/W | Reset: 0x00770000 (0bxxxxxxxxxx111x111xx000000xx000000)

Bit	Reset	Description
22:20	0x7	AUTO_CAL_DQS_TERM_SLOPE: [PMC] slope for DQS term value - range is 0.125 to 1.000 in steps of 0.125
18:16	0x7	AUTO_CAL_DQ_TERM_SLOPE: [PMC] slope for DQ term value - range is 0.125 to 1.000 in steps of 0.125
13:8	0x0	AUTO_CAL_DQS_TERM_OFFSET: [PMC] 2's complement offset for DQS term value
5:0	0x0	AUTO_CAL_DQ_TERM_OFFSET: [PMC] 2's complement offset for DQ term value

ZQ Calibration Registers

1. Periodic mode

   Allows a ZQ calibration command to be sent periodically to DRAM. After ZCAL_INTERVAL, ZCAL_MRW_CMD will be sent to each DRAM, 1 at a time, followed by ZCAL_WAIT_CNT interval in which no other commands will be allowed to be sent to either DRAM. So if ZQ_MRW_DEV_SELECTN == 2'b00, it would send the MRW command to dev0, wait ZCAL_WAIT_CNT, send the command to dev1, wait ZCAL_WAIT_CNT, resume normal operation. It will then wait

ZCAL_INTERVAL microseconds before repeating the procedure. The ZQ calibration command will not be sent to devices that are 1) unpopulated and 2) either in or about to enter self-refresh or DPD.

To enable periodic ZQ calibration, program ZCAL_INTERVAL and ZCAL_WAIT_CNT to be non-zero, as well as ZCAL_MRW_CMD to contain the command and device selection, followed by TIMING_CONTROL to latch those three registers.

Note that ZCAL_WAIT_CNT will be used instead of MRS_WAIT_CNT for delaying the subsequent DRAM commands after the ZQ calibration commands, even if a ZQ calibration command is also an MRW command.

Disable this feature by setting ZCAL_INTERVAL.ZCAL_REF_INTERVAL = 0, followed by TIMING_CONTROL to latch it.

2. One-shot mode

   Allows ZQ calibration command to be sent to DRAM.

   To send a one-shot command, keep ZCAL_INTERVAL and ZCAL_WAIT_CNT equal to zero, program ZCAL_MRW_CMD with only one device selected, then program TIMING_CONTROL to latch in those registers, and finally program ZQ_CAL_CMD to 1 to trigger the one-shot command. If more than one device needs to be calibrated, wait the ZQ calibration time and repeat the steps above with the other device selection bit enabled in ZCAL_MRW_CMD.

   Do not send a one-shot ZQ calibration command when periodic mode is enabled.

ZQ Calibration Registers – LPDDR4

1. Periodic mode

   Allows a ZQ calibration start command to be sent periodically to DRAM. LPDDR4 requires a separate MPC command to latch the ZQ calibration results. So, the LPDDR4 sequence looks like this for each DRAM:

   - after ZCAL_INTERVAL, an MPC ZQCal Start command will be sent
   - after ZCAL_WAIT_CNT interval, an MPC ZQCal Latch command will be sent
   - after ZCAL_LATCH_CNT interval, subsequent ZQ calibration of additional DRAM occurs or other commands can be issued

   So if ZQ_MRW_DEV_SELECTN == 2'b00, it would send the Start command to dev0, wait ZCAL_WAIT_CNT, send Latch command to dev0, wait ZCAL_LATCH_CNT, send Start command to dev1, wait ZCAL_WAIT_CNT, send Latch command to dev1, wait ZCAL_LATCH_CNT, resume normal operation. It will then wait ZCAL_INTERVAL microseconds before repeating the procedure. The ZQ calibration command will not be sent to devices that are 1) unpopulated and 2) either in or about to enter self-refresh or DPD.

   To enable periodic ZQ calibration:

   - enable FBIO_CFG7.ZQCAL_MPC_ENABLE and FBIO_CFG7.ZQCAL_LATCH_ENABLE
   - program ZCAL_INTERVAL, ZCAL_WAIT_CNT, and ZCAL_LATCH_CNT to be non-zero
   - program ZCAL_MRW_CMD.ZQ_MRW_OP=ZQCal_Start and ZCAL_MRW_CMD.ZQ_MRW_MA=ZQCal_Latch
   - program ZQ_MRW_DEV_SELECTN to contain the device selection
   - write to TIMING_CONTROL to latch the three registers

   To disable periodic ZQ calibration, program ZCAL_INTERVAL.ZCAL_REF_INTERVAL = 0, followed by TIMING_CONTROL to latch it.

2. One-shot mode

   Allows ZQ calibration command to be sent to DRAM.

   To send a one-shot command, keep ZCAL_INTERVAL and ZCAL_WAIT_CNT equal to zero, program ZCAL_MRW_CMD with only one device selected, then program TIMING_CONTROL to latch in those registers, and finally program ZQ_CAL_CMD to 1 to trigger the one-shot command. Wait the ZQ calibration time and program ZQ_LATCH_CMD to 1. If more than one device needs to be calibrated, wait the ZQ calibration time and repeat the steps above with the other device selection bit enabled in ZCAL_MRW_CMD. Only one of ZQ_CAL_CMD and ZQ_LATCH_CMD should be 1 for any ZQ_CAL write.

Do not send a one-shot ZQ calibration command when periodic mode is enabled.

18.11.2.95 EMC_ZCAL_INTERVAL_0

Configure ZQ Calibration

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:10	0x0	ZCAL_INTERVAL_HI: [PMC] Combined with ZCAL_INTERVAL_LO specifies the number of microseconds to wait between issuance of ZCAL_MRW_CMD. If 0, ZCAL is disabled and internal counter will be reset.
9:0	0x0	ZCAL_INTERVAL_LO [PMC]

18.11.2.96 EMC_ZCAL_WAIT_CNT_0

Configure ZQ Calibration

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the
- Boot ROM during warm boot.

Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxx000000xxxxx00000000000)

Bit	Reset	Description
31	DISABLED	ZCAL_RESISTOR_SHARED: [PMC] ENABLED is ZQ resistor is shared across ranks 0 = DISABLED 1 = ENABLED
21:16	0x0	ZCAL_LATCH_CNT: [PMC] LPDDR4 - Number of EMC clocks to wait before issuing any commands after sending MPC ZQCAL_LATCH command.
10:0	0x0	ZCAL_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending ZCAL_MRW_CMD. LPDDR4 - Number of EMC clocks to wait before issuing any commands after sending MPC ZQCAL_START command.

18.11.2.97 EMC_ZCAL_MRW_CMD_0

Configure ZQ Calibration

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e8 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x3	ZQ_MRW_DEV_SELECTN: [PMC] Active-low chip-select, 0x0 applies command to both devices (will happen 1 at a time), 0x2 to for only dev0, 0x1 for dev1.

Bit	Reset	Description
23:16	0x0	ZQ_MRW_MA: [PMC] MRW MA field to be sent after ZCAL_INTERVAL. LPDDR4 - program to ZQCal_Latch
7:0	0x0	ZQ_MRW_OP: [PMC] MRW OP field to be sent after ZCAL_INTERVAL. LPDDR4 - program to ZQCal_Start

18.11.2.98 EMC_ZQ_CAL_0

Trigger a Single ZQ Calibration

This register issues a ZQ calibration MRS command.

Offset: 0x2ec | Read/Write: R/W | Reset: 0x00000010 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxx1xx00)

Bit	Reset	Description
31:30	0x0	ZQ_CAL_DEV_SELECTN: Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1, 0x3 for neither device (0x0, for both devices, is not allowed if the ZQ resister is shared between devices).
4	LONG	ZQ_CAL_LENGTH: Indicate short or long ZQ calibration. 0 = SHORT 1 = LONG
1	0x0	ZQ_LATCH_CMD: Issues a ZQ latch command (LPDDR4 only).
0	0x0	ZQ_CAL_CMD: Issues a ZQ calibration command (DDR3 only).

18.11.2.99 EMC_XM2COMPPADCTRL3_0

Autocal COMP pad control register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
17	DISABLE	EMC2TMC_CFG_XM2COMP_PD_E_WKPD: [PMC] Enable weak pull-down 0 = DISABLE 1 = ENABLE
16	DISABLE	EMC2TMC_CFG_XM2COMP_PD_E_WKPU: [PMC] Enable weak pull-up 0 = DISABLE 1 = ENABLE
15:14	0x0	EMC2TMC_CFG_XM2COMP_PD_DRVDN_ZCTRL: [PMC] Drive pull-down impedance calibration select. 0x0 = 40ohms, 0x1 = 60ohms, 0x2 = 80ohms, 0x3 = 120ohms
13:12	0x0	EMC2TMC_CFG_XM2COMP_PD_DRVUP_ZCTRL: [PMC] Drive pull-up impedance calibration select. 0x0 = 40ohms, 0x1 = 60ohms, 0x2 = 80ohms, 0x3 = 120ohms
11:10	0x0	EMC2TMC_CFG_XM2COMP_PD_VDDA_MODE: [PMC] Function regulator rail from adjacent IOBRICK. 11 to enable local switch to have better power delivery
9:8	0x0	EMC2TMC_CFG_XM2COMP_PD_VAUXP_MODE: [PMC] Function regulator rail from adjacent IOBRICK. 11 to enable local switch to have better power delivery
7:6	0x0	EMC2TMC_CFG_XM2COMP_PD_VCLAMP_MODE: [PMC] Function regulator rail from adjacent IOBRICK. 11 to enable local switch to have better power delivery
5:2	0x0	EMC2TMC_CFG_XM2COMP_PD_TEST_MODE: [PMC] Test mode select
1:0	SCHMITT	EMC2TMC_CFG_XM2COMP_PD_RX_MODE: [PMC] Set front-end RX type 0 = SCHMITT 1 = RFU0 2 = DIFFAMP 3 = RFU1

18.11.2.100 EMC_AUTO_CAL_VREF_SEL_0_0

Autocal master measure control register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2f8 | Read/Write: R/W | Reset: 0x3c3c3c3c (0b00111100001111000011110000111100)

Bit	Reset	Description
31	DISABLE	AUTO_CAL_PD_VREF1_CAL_MODE: [PMC] IVREF_CAL_MODE for vref1 pull-down calibration measure cycle 0 = DISABLE 1 = ENABLE
30:24	0x3c	AUTO_CAL_PD_VREF1_SEL: [PMC] IVREF_LVL for vref1 pull-down calibration measure cycle
23	DISABLE	AUTO_CAL_PU_VREF1_CAL_MODE: [PMC] IVREF_CAL_MODE for vref1 pull-up calibration measure cycle 0 = DISABLE 1 = ENABLE
22:16	0x3c	AUTO_CAL_PU_VREF1_SEL: [PMC] IVREF_LVL for vref1 pull-up calibration measure cycle
15	DISABLE	AUTO_CAL_PD_VREF0_CAL_MODE: [PMC] IVREF_CAL_MODE for vref0 pull-down calibration measure cycle 0 = DISABLE 1 = ENABLE
14:8	0x3c	AUTO_CAL_PD_VREF0_SEL: [PMC] IVREF_LVL for vref0 pull-down calibration measure cycle
7	DISABLE	AUTO_CAL_PU_VREF0_CAL_MODE: [PMC] IVREF_CAL_MODE for vref0 pull-up calibration measure cycle 0 = DISABLE 1 = ENABLE
6:0	0x3c	AUTO_CAL_PU_VREF0_SEL: [PMC] IVREF_LVL for vref0 pull-up calibration measure cycle

18.11.2.101 EMC_AUTO_CAL_VREF_SEL_1_0

Autocal master measure control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x300 | Read/Write: R/W | Reset: 0x00003c3c (0bxxxxxxxxxxxxxxxx0011110000111100)

Bit	Reset	Description
15	DISABLE	AUTO_CAL_PD_VREF2_CAL_MODE: [PMC] IVREF_CAL_MODE for vref2 pull-down calibration measure cycle 0 = DISABLE 1 = ENABLE
14:8	0x3c	AUTO_CAL_PD_VREF2_SEL: [PMC] IVREF_LVL for vref2 pull-down calibration measure cycle
7	DISABLE	AUTO_CAL_PU_VREF2_CAL_MODE: [PMC] IVREF_CAL_MODE for vref2 pull-up calibration measure cycle 0 = DISABLE 1 = ENABLE
6:0	0x3c	AUTO_CAL_PU_VREF2_SEL: [PMC] IVREF_LVL for vref2 pull-up calibration measure cycle

18.11.2.102 EMC_XM2COMPPADCTRL_0

Autocal COMP pad control register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x30c | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxx000000000x0xx0110x00)

Bit	Reset	Description
19:12	0x0	EMC2TMC_CFG_XM2COMP_RFU: [PMC] Reserved
11	DISABLE	EMC2TMC_CFG_XM2COMP_E_TESTOUT: [PMC] ENABLE = Select test output (see TEST_MODE for available options) 0 = DISABLE 1 = ENABLE
9	DISABLE	EMC2TMC_CFG_XM2COMP_E_PWRD: [PMC] Active High to disable DC current paths within entire cell DISABLE for normal operation and ENABLE for IDDQ testing 0 = DISABLE 1 = ENABLE
6:4	0x3	EMC2TMC_CFG_XM2COMP_BG_SETUP: [PMC] Temperature coefficient pin
3	HIGHPERF	EMC2TMC_CFG_XM2COMP_BG_MODE: [PMC] Set regulator mode 0 = HIGHPERF 1 = LOWPOWER
1:0	LPDDR23	EMC2TMC_CFG_XM2COMP_MEM_MODE: [PMC] Type of DDR memory used for misc. internal controls 0 = LPDDR23 1 = SDDR3 2 = LPDDR4 3 = RFU

18.11.2.103 EMC_FDPD_CTRL_DQ_0

Configures DPD settle times

Offset: 0x310 | Read/Write: R/W | Reset: 0x8020221f (0b10x000000010xxx00010001000011111)

Bit	Reset	Description
31:30	0x2	DQ_PHASE_MIN: [PMC] lower limiter of the DPD phase
28:24	0x0	DQ_DPD_EXIT_DELAY: [PMC] exit delay to cmd. (This is the settle delay of the last phase to cmd, needs to factor in write latency.)
23:20	0x2	DQ_PHASE_RAMP_IN_TIME: [PMC] time of each phase when ramping into FDPD state
16:12	0x2	DQ_PHASE_RAMP_OUT_TIME: [PMC] time of each phase when ramping out of FDPD state
11:8	0x2	DQ_PHASE_HOLD_MIN: [PMC] minimal time a DPD phase is hold
7:0	0x1f	DQ_DPD_ENTRY_DELAY: [PMC] initial delay after idle before starting the FDPD ramp process

18.11.2.104 EMC_FDPD_CTRL_CMD_0

Configures DPD settle times

Offset: 0x314 | Read/Write: R/W | Reset: 0x0220f40f (0b000000100010xxx01111010000001111)

Bit	Reset	Description
31:30	0x0	CMD_PHASE_MIN: [PMC] lower limiter of the DPD phase
29	0x0	CMD_DPD_RAMP_ENABLE: [PMC] Enables the ramp feature on cmd.
28:24	0x2	CMD_DPD_EXIT_DELAY: [PMC] exit delay to cmd. (This is the settle delay of the last phase to cmd, needs to factor in wl)
23:20	0x2	CMD_PHASE_RAMP_IN_TIME: [PMC] time of each phase when ramping into FDPD state
16:12	0xf	CMD_PHASE_RAMP_OUT_TIME: [PMC] time of each phase when ramping out of FDPD state

Bit	Reset	Description
11:8	0x4	CMD_PHASE_HOLD_MIN: [PMC] minimal time a DPD phase is hold
7:0	0xf	CMD_DPD_ENTRY_DELAY: [PMC] initial delay after idle before starting the FDPD ramp process

18.11.2.105 EMC_PMACRO_CMD_BRICK_CTRL_FDPD_0

Controls the RFU pin of the IOBRICK on cmd channels

Offset: 0x318 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17:16	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT8: [PMC]
15:14	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT7: [PMC]
13:12	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT6: [PMC]
11:10	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT5: [PMC]
9:8	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT4: [PMC]
7:6	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT3: [PMC]
5:4	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT2: [PMC]
3:2	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT1: [PMC]
1:0	0x0	CMD_BRICK_CTRL_FDPD_RFU_PHASE_BIT0: [PMC]

18.11.2.106 EMC_PMACRO_DATA_BRICK_CTRL_FDPD_0

Controls the RFU pin of the IOBRICK on data channels

Offset: 0x31c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17:16	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT8: [PMC] assigned to dqs_tx_pwrd
15:14	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT7: [PMC] assinged to dq[8:5] tx_pwrd
13:12	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT6: [PMC] assigned to dq[4:0] tx_pwrd
11:10	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT5: [PMC] assigned to RFU[13] as spare
9:8	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT4: [PMC] assigned to RFU[12] DQS VAUXC domain flops
7:6	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT3: [PMC] assigned to RFU[11] DO VAUXC domain flops
5:4	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT2: [PMC] assigned to RFU[10:9] dqsp/n short trimmer
3:2	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT1: [PMC] assigned to RFU[8:5] dq[8:5] short trimmer
1:0	0x0	DATA_BRICK_CTRL_FDPD_RFU_PHASE_BIT0: [PMC] assigned to RFU[4:0] dq[4:0] short trimmer

18.11.2.107 EMC_SCRATCH0_0

This is a scratch register for general use.

Offset: 0x324 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH:

18.11.2.108 EMC_PMACRO_BRICK_CTRL_RFU1_0

Controls the lower 16 RFU pin of the IOBRICK (CMD and DATA)

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x330 | Read/Write: R/W | Reset: 0x1fff1fff (0b00011111111111111000111111111111)

Bit	Reset	Description
31:16	0x1fff	DATA_BRICK_CTRL_RFU1: [PMC]
15:0	0x1fff	CMD_BRICK_CTRL_RFU1: [PMC]

18.11.2.109 EMC_PMACRO_BRICK_CTRL_RFU2_0

Controls the upper 16 RFU pin of the IOBRICK (CMD and DATA)

Offset: 0x334 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DATA_BRICK_CTRL_RFU2: [PMC]
15:0	0x0	CMD_BRICK_CTRL_RFU2: [PMC]

18.11.2.110 EMC_CMD_MAPPING_CMD0_0_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x380 | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD0_DQ3_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 3 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD0_DQ2_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 2 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD0_DQ1_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 1 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD0_DQ0_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 0 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.111 EMC_CMD_MAPPING_CMD0_1_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x384 | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD0_DQ7_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 7 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD0_DQ6_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 6 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD0_DQ5_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 5 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD0_DQ4_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 4 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.112 EMC_CMD_MAPPING_CMD0_2_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x388 | Read/Write: R/W | Reset: 0x01010101 (0bxxxx0001x0000001x0000001x0000001)

Bit	Reset	Description
27:24	MAP_BOOT	CMD0_DQ_CMD_MAP: [PMC_SECURE] Mapping of CMD brick 0, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1

Bit	Reset	Description
22:16	MAP_BOOT	CMD0_DQSN_MAP: [PMC_SECURE] Mapping of CMD brick 0, DQSN pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
14:8	MAP_BOOT	CMD0_DQSP_MAP: [PMC_SECURE] Mapping of CMD brick 0, DQSP pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
6:0	MAP_BOOT	CMD0_DQ8_MAP: [PMC_SECURE] Mapping of CMD brick 0, data pin 8 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.113 EMC_CMD_MAPPING_CMD1_0_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x38c | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD1_DQ3_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 3 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD1_DQ2_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 2 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD1_DQ1_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 1 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD1_DQ0_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 0 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.114 EMC_CMD_MAPPING_CMD1_1_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warmboot.

Offset: 0x390 | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD1_DQ7_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 7 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD1_DQ6_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 6 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD1_DQ5_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 5 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD1_DQ4_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 4 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.115 EMC_CMD_MAPPING_CMD1_2_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x394 | Read/Write: R/W | Reset: 0x01010101 (0bxxxx0001x0000001x0000001x0000001)

Bit	Reset	Description
27:24	MAP_BOOT	CMD1_DQ_CMD_MAP: [PMC_SECURE] Mapping of CMD brick 1, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1

Bit	Reset	Description
22:16	MAP_BOOT	CMD1_DQSN_MAP: [PMC_SECURE] Mapping of CMD brick 1, DQSN pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
14:8	MAP_BOOT	CMD1_DQSP_MAP: [PMC_SECURE] Mapping of CMD brick 1, DQSP pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
6:0	MAP_BOOT	CMD1_DQ8_MAP: [PMC_SECURE] Mapping of CMD brick 1, data pin 8 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.116 EMC_CMD_MAPPING_CMD2_0_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x398 | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD2_DQ3_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 3 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD2_DQ2_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 2 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD2_DQ1_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 1 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD2_DQ0_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 0 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.117 EMC_CMD_MAPPING_CMD2_1_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x39c | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD2_DQ7_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 7 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD2_DQ6_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 6 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD2_DQ5_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 5 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD2_DQ4_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 4 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.118 EMC_CMD_MAPPING_CMD2_2_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3a0 | Read/Write: R/W | Reset: 0x01010101 (0bxxxx0001x0000001x0000001x0000001)

Bit	Reset	Description
27:24	MAP_BOOT	CMD2_DQ_CMD_MAP: [PMC_SECURE] Mapping of CMD brick 2, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1

Bit	Reset	Description
22:16	MAP_BOOT	CMD2_DQSN_MAP: [PMC_SECURE] Mapping of CMD brick 2, DQSN pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
14:8	MAP_BOOT	CMD2_DQSP_MAP: [PMC_SECURE] Mapping of CMD brick 2, DQSP pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
6:0	MAP_BOOT	CMD2_DQ8_MAP: [PMC_SECURE] Mapping of CMD brick 2, data pin 8 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.119 EMC_CMD_MAPPING_CMD3_0_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3a4 | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD3_DQ3_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 3 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD3_DQ2_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 2 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD3_DQ1_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 1 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD3_DQ0_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 0 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.120 EMC_CMD_MAPPING_CMD3_1_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3a8 | Read/Write: R/W | Reset: 0x01010101 (0bx0000001x0000001x0000001x0000001)

Bit	Reset	Description
30:24	MAP_BOOT	CMD3_DQ7_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 7 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
22:16	MAP_BOOT	CMD3_DQ6_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 6 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
14:8	MAP_BOOT	CMD3_DQ5_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 5 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

Bit	Reset	Description
6:0	MAP_BOOT	CMD3_DQ4_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 4 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.121 EMC_CMD_MAPPING_CMD3_2_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3ac | Read/Write: R/W | Reset: 0x01010101 (0bxxxx0001x0000001x0000001x0000001)

Bit	Reset	Description
27:24	MAP_BOOT	CMD3_DQ_CMD_MAP: [PMC_SECURE] Mapping of CMD brick 3, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1

Bit	Reset	Description
22:16	MAP_BOOT	CMD3_DQSN_MAP: [PMC_SECURE] Mapping of CMD brick 3, DQSN pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
14:8	MAP_BOOT	CMD3_DQSP_MAP: [PMC_SECURE] Mapping of CMD brick 3, DQSP pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1 37 = MAP_CLK 38 = MAP_CLKB

Bit	Reset	Description
6:0	MAP_BOOT	CMD3_DQ8_MAP: [PMC_SECURE] Mapping of CMD brick 3, data pin 8 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_ADR0 3 = MAP_ADR1 4 = MAP_ADR2 5 = MAP_ADR3 6 = MAP_ADR4 7 = MAP_ADR5 8 = MAP_ADR6 9 = MAP_ADR7 10 = MAP_ADR8 11 = MAP_ADR9 12 = MAP_ADR10 13 = MAP_ADR11 14 = MAP_ADR12 15 = MAP_ADR13 16 = MAP_ADR14 17 = MAP_ADR15 18 = MAP_SUB0 19 = MAP_SUB1 20 = MAP_SUB2 21 = MAP_BA0 22 = MAP_BA1 23 = MAP_BA2 24 = MAP_CAS 25 = MAP_RAS 26 = MAP_WE 27 = MAP_CS0 28 = MAP_CS1 29 = MAP_CS_B0 30 = MAP_CS_B1 31 = MAP_ODT0 32 = MAP_ODT1 33 = MAP_ODT_B0 34 = MAP_ODT_B1 35 = MAP_GPIO0 36 = MAP_GPIO1

18.11.2.122 EMC_CMD_MAPPING_BYTE_0

Configures the command mapping

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3b0 | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	MAP_BOOT	BYTE7_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE7 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1

Bit	Reset	Description
27:24	MAP_BOOT	BYTE6_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE6 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1
23:20	MAP_BOOT	BYTE5_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE5 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1
19:16	MAP_BOOT	BYTE4_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE4 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1
15:12	MAP_BOOT	BYTE3_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE3 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1

Bit	Reset	Description
11:8	MAP_BOOT	BYTE2_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE2 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1
7:4	MAP_BOOT	BYTE1_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE1 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1
3:0	MAP_BOOT	BYTE0_DQ_CMD_MAP: [PMC_SECURE] Mapping of BYTE0 brick, CMD pin 0 = MAP_NONE 1 = MAP_BOOT 2 = MAP_RESET 3 = MAP_CH0_CKE0 4 = MAP_CH0_CKE1 5 = MAP_CH0_CKE_B0 6 = MAP_CH0_CKE_B1 7 = MAP_CH1_CKE0 8 = MAP_CH1_CKE1 9 = MAP_CH1_CKE_B0 10 = MAP_CH1_CKE_B1 11 = MAP_ODT0 12 = MAP_ODT1 13 = MAP_ODT_B0 14 = MAP_ODT_B1

18.11.2.123 EMC_TR_TIMING_0_0

Offset: 0x3b4 | Read/Write: R/W | Reset: 0x011861b8 (0bxxxxx0010x0110xx0110xx0110111000)

Bit	Reset	Description
26:23	0x2	T_CATR_RD2VREF: CATR Read to CAVREF
21:18	0x6	T_DHTRAIN: Hold time at DQS toggle for CAVREF capture
15:12	0x6	T_DSTRAIN: Setup time at DQS toggle for CAVREF capture
9:0	0x1b8	T_VREF_LONG: Delay between CAVREF and CATR Reads

18.11.2.124 EMC_TR_CTRL_0_0

Offset: 0x3b8 | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxxxxxxx001000000)

Bit	Reset	Description
8	0x0	CATR_ENABLE: Indicator to EMC that CA Trainig is enabled so that EMC treats CKE high/low differently for CA Training.

Bit	Reset	Description
7:5	0x2	CAVREF_DQS_DURATION: No of DQS toggle to latch CAVREF
4	0x0	CAVREF_DRIVE_DQS: Drive DQS
3:0	0x0	CAVREF_TX_BYTE_MASK: Mask TX_D/EN of DQ/DQS during CAVREF (Lower byte of Subp needs to have DQS/DQ driven, upper should be masked)

18.11.2.125 EMC_TR_CTRL_1_0

Offset: 0x3bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	ENABLE_TR_QRST: Select TR QRST
3	0x0	ENABLE_TR_QSAFE: Select TR QSAFE
2	0x0	ENABLE_TR_RDV_MASK: Select TR RDV_MASK
1	0x0	ENABLE_TR_QPOP: Select TR QPOP
0	0x0	ENABLE_TR_RDV: Select TR RDV

18.11.2.126 EMC_SWITCH_BACK_CTRL_0

Triggers clock switchback request from EMC to CAR.

Offset: 0x3c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CLK_SWITCH_BACK

18.11.2.127 EMC_TR_RDV_0

RDV used for training

Offset: 0x3c4 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0001111)

Bit	Reset	Description
6:0	0xf	TR_RDV: 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.128 EMC_STALL_THEN_EXE_BEFORE_CLKCHANGE_0

Scheme to change controller timing parameters + SDRAM mode timing atomically.

1. Program the EMC controller timing registers (shadowed).
2. Pre-program the SDRAM mode registers.
   a. Write 1 to the STALL_THEN_EXE_BEFORE_CLKCHANGE field, which will prevent further cfg execution until a clock change request has been detected.

Note: *After this step, no register read should happen on the EMC before the clock change (C) happens, otherwise the system will hang. Disable the EMC interrupt before this step if the interrupt handler reads EMC registers.*

---

   b. Write to whatever EMC registers you want to be execute after clock change request has been detected but before the actual clock change happens. For example, if you want to disable DLL in DDR3 mode, you can,
      - Issue a write to SDRAM's MR1 register via EMC MRS register to disable DLL.
      - Issue a Self-refresh entry via EMC SELF_REF register.

Note: *If there is no need to execute any register access before clock change, skip step (b); step (a) is still required.*

c. Write 1 to STALL_THEN_EXE_AFTER_CLKCHANGE field which will prevent further config execution until after the actual clock change has happened.

d. Write to whatever EMC registers you want to be executed after the clock change. Using the same example as (b), you will,

- Issue a Self-refresh exit via EMC SELF_REF register.
- Issue writes to SDRAM's MRx register(s) to change read/write latencies and/or enable DLL
- Issue burst refreshes via EMC REF register.

Note: *If there is no need to execute any register access after clock change, skip step (d); step (c) is still required.*

3. Program the CAR to change either clock source and/or clock divider.

Notes:

- *In both sequences above, DYN_SELF_REF must be disabled off before the first step, then it can be restored after the last step.*
- *The interval between two clock change sequences are recommended to be at least 20 µs apart.*

Writes to this register will stall the register read/write path until a clock change request is detected. Once detected, whatever register access follows this register write will be executed until a STALL_THEN_EXE_AFTER_CKLCHANGE is encountered. At that point, EMC timing registers will be updated and the clock change request will be acknowledged.

Offset: 0x3c8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	STALL_THEN_EXE_BEFORE_CLKCHANGE: Writing a one to this bit will stall subsequent register accesses.

18.11.2.129 EMC_STALL_THEN_EXE_AFTER_CLKCHANGE_0

Writes to this register will stall the register read/write path until after EMC timing registers are updated and that clock change has been completed. Once that happens, whatever register access follows this register write will be executed until UNSTALL_RW_AFTER_CLKCHANGE is encountered. At that point, normal memory read/write will be allowed to resume.

Offset: 0x3cc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	STALL_THEN_EXE_AFTER_CLKCHANGE: Writing a one to this bit will stall subsequent register accesses.

18.11.2.130 EMC_UNSTALL_RW_AFTER_CLKCHANGE_0

Writing to this register will unstall memory reads/writes after a clock change. Use in conjunction with STALL_THEN_EXE_AFTER_CLKCHANGE.

Offset: 0x3d0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UNSTALL_RW_AFTER_CLKCHANGE: Writing a one to this bit will unstall memory reads/writes after a clock change.

18.11.2.131 EMC_AUTO_CAL_ STATUS2_0

Autocal master status register

Offset: 0x3d4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:24	X	AUTO_CAL_VREF2_DRVDN: Pull-down code from VREF2 measure cycle.
21:16	X	AUTO_CAL_VREF2_DRVUP: Pull-up code from VREF2 measure cycle.
13:8	X	AUTO_CAL_VREF1_DRVDN: Pull-down code from VREF1 measure cycle.
5:0	X	AUTO_CAL_VREF1_DRVUP: Pull-up code from VREF1 measure cycle.

18.11.2.132 EMC_SEL_DPD_CTRL_0

Configures Functional SEL_DPD Modes

Offset: 0x3d8 | Read/Write: R/W | Reset: 0x00040000 (0bxxxxxxxxxxxxxx100xxxxxxx0xx0000xx)

Bit	Reset	Description
18:16	0x4	SEL_DPD_DLY: [PMC] number of cycles to wait before asserting SEL_DPD
8	DISABLED	DATA_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for data pads 0 = DISABLED 1 = ENABLED
5	DISABLED	ODT_SEL_DPD_EN: [PMC] tie SEL_DPD for ODT pads to ENABLE_ODT_DURING_WRITE 0 = DISABLED 1 = ENABLED
4	DISABLED	RESET_SEL_DPD_EN: [PMC] assert SEL_DPD for reset pad 0 = DISABLED 1 = ENABLED
3	DISABLED	CA_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for discrete command/address pads 0 = DISABLED 1 = ENABLED
2	DISABLED	CLK_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for discrete clock pad 0 = DISABLED 1 = ENABLED

18.11.2.133 EMC_PRE_REFRESH_REQ_CNT_0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x3dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	PRE_REF_REQ_CNT: [PMC] When the refresh counter reach this pre-refresh request count before a burst refresh will be issued when it reach the {REFRESH,REFRESH_LO}, a pre-refresh request will be asserted to MC to close the banks/pages and flush its pipeline. A 0 value will disable this feature.

18.11.2.134 EMC_DYN_SELF_REF_CONTROL_0

Threshold for Dynamic Self-Refresh Entry

This register controls how dynamic self-refresh entry/exit is handled.

If ODT is enabled, the DSR_PER_DEVICE field must be set to DISABLED.

It is recommended to change the values of ODT_WRITE and DSR_PER_DEVICE with a clock change

To do it outside of a clock change sequence, one has to be aware that DSR enable/disable takes time to take effect. This is the sequence for disabling DSR_PER_DEVICE and enabling ODT outside of clock change:

1. DYN_SELF_REF <= DISABLED (DSR exit)

2. TIMING_UPDATE <= 1 (latch in the shadow value)
3. Read back and discard any EMC register such as INTSTATUS (ensure 1 and 2 have gone through)
4. Wait 2 µs for the last possible self-refresh entry
5. Poll for SDRAM_IN_SELF_REFRESH == 0 (ensure that self-refresh exits)
6. ODT_WRITE <= new value to turn on ODT

   DSR_PER_DEVICE <= DISABLED (turn off per-device DSR)

   DYN_SELF_REF <= ENABLED (re-enable DSR)
7. TIMING_UPDATE <= 1 (latch in the shadow values)

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x3e0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	DISABLED	DSR_PER_DEVICE: [PMC] controls whether dynamic self-refresh is done on a per-device basis. PER_DEVICE_DSR must be enabled in single-rank system also. 0 = DISABLED 1 = ENABLED
15:0	0x0	DSR_THRESHOLD: [PMC] number of idle cycles to wait before allowing dynamic self-refresh entry

18.11.2.135 EMC_TXSRDLL_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the
- Boot ROM during warm boot.

Note: If operating in non-DLL mode, this register needs to be updated with the non-DLL timing requirement.

Offset: 0x3e4 | Read/Write: R/W | Reset: 0x000007ff (0bxxxxxxxxxxxxxxxxxxxx011111111111)

Bit	Reset	Description
11:0	0x7ff	TXSRDLL: [PMC] Cycles between self-refresh exit and first DRAM command requiring a locked DLL. For DDR3 only: READ (and RAP) and synchronous ODT commands. \ Largest allowed value is 0xffe

18.11.2.136 EMC_CCFIFO_ADDR_0

CCFIFO_ADDR: This register contains the address offset of the EMC register that is intended to be executed during the clock change sequence.

> ***Note:*** *Before triggering a clock change sequence from the CAR, you must configure the pre-/post-clock change sequence by writing, multiple times, to the CCFIFO_DATA/CCFIFO_ADDR register pair. After the CCFIFO_ADDR register is written, Hardware will write the register offset and corresponding data into the 32-deep clock change FIFO (CCFIFO).*

Offset: 0x3e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CCFIFO_STALL_CNT_BY_1:if set, count by 1 instead of 256 clk increments

Bit	Reset	Description
30:16	0x0	CCFIFO_STALL_CNT: non-zero value will stall CCFIFO processing counter uses a hardcoded 8-bit pre-scalar; e.g. CNT=1 results in a 256 clock delay or 1 if CCFIFO_STALL_CNT_BY_1 is set
15:0	0x0	CCFIFO_ADDR: clock change FIFO address register.

18.11.2.137 EMC_CCFIFO_DATA_0

CCFIFO_DATA: This register contains the 32-bit data of the EMC register that is intended to be written to or pointed at by CCFIFO_ADDR.

Offset: 0x3ec | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CCFIFO_DATA: clock change FIFO data register.

18.11.2.138 EMC_CCFIFO_STATUS_0

Offset: 0x3f0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:0	X	CCFIFO_COUNT

18.11.2.139 EMC_TR_QPOP_0

DRAM timing parameter

Offset: 0x3f4 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxx0000000xxxxxxxx0000110)

Bit	Reset	Description
22:16	0x0	TR_QPOP_PREAMBLE:time from read command to pop preamble from the pad macro FIFO. EMC ignores this field if FBIO_CFG7.QPOP_RD_PREAMBLE_TOGGLE=0. 60 = MAX_1TO1 120 = MAX_2TO1
6:0	0x6	TR_QPOP: Time from training read command to pop data from the pad macro FIFO. 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.140 EMC_TR_RDV_MASK_0

DRAM timing parameter

Offset: 0x3f8 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000110)

Bit	Reset	Description
6:0	0x6	TR_RDV_MASK: Serves the same purpose as RDV_MASK serves for RDV. Training needs separate RDV so a separate RDV_MASK. 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.141 EMC_TR_QSAFE_0

DRAM timing parameter

Offset: 0x3fc | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000110)

Bit	Reset	Description
6:0	0x6	TR_QSAFE: Training time from a read command to when it is safe to issue a QRST (delayed by the QRST parameter). 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.142 EMC_TR_QRST_0

DRAM timing parameter

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxx00000xxxxxxxxx0000110)

Bit	Reset	Description
20:16	0x0	TR_QRST_DURATION: QRST remains asserted for QRST_DURATION + 1 cycles
6:0	0x6	TR_QRST: Training time from a read command to when it is safe to issue a QRST (delayed by the QRST parameter). 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.143 EMC_SWIZZLE_RANK0_BYTE0_0

The SWZ_RANK*_BYTE*_BIT*_SEL fields indicate which SDRAM data bit (0-7) within each byte is mapped/connected to the corresponding bit of the chip.

Offset: 0x404 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE0_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE0_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE0_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE0_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE0_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE0_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE0_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE0_BIT0_SEL: [PMC]

18.11.2.144 EMC_SWIZZLE_RANK0_BYTE1_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE1_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE1_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE1_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE1_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE1_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE1_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE1_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE1_BIT0_SEL: [PMC]

18.11.2.145 EMC_SWIZZLE_RANK0_BYTE2_0

Offset: 0x40c | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE2_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE2_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE2_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE2_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE2_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE2_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE2_BIT1_SEL: [PMC]

Bit	Reset	Description
2:0	0x0	SWZ_RANK0_BYTE2_BIT0_SEL: [PMC]

18.11.2.146 EMC_SWIZZLE_RANK0_BYTE3_0

Offset: 0x410 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE3_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE3_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE3_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE3_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE3_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE3_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE3_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE3_BIT0_SEL: [PMC]

18.11.2.147 EMC_SWIZZLE_RANK1_BYTE0_0

Offset: 0x418 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE0_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE0_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE0_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE0_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE0_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE0_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE0_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE0_BIT0_SEL: [PMC]

18.11.2.148 EMC_SWIZZLE_RANK1_BYTE1_0

Offset: 0x41c | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE1_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE1_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE1_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE1_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE1_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE1_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE1_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE1_BIT0_SEL: [PMC]

18.11.2.149 EMC_SWIZZLE_RANK1_BYTE2_0

Offset: 0x420 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE2_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE2_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE2_BIT5_SEL: [PMC]

Bit	Reset	Description
18:16	0x4	SWZ_RANK1_BYTE2_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE2_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE2_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE2_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE2_BIT0_SEL: [PMC]

18.11.2.150 EMC_SWIZZLE_RANK1_BYTE3_0

Offset: 0x424 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE3_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE3_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE3_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE3_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE3_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE3_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE3_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE3_BIT0_SEL: [PMC]

18.11.2.151 EMC_ISSUE_QRST_0

DRAM timing parameter

Offset: 0x428 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ISSUE_QRST: Sets QRST to pads/macros. must be cleared to resume operations

18.11.2.152 EMC_PMC_SCRATCH1_0

Offset: 0x440 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH1:[PMC] Scratch register for E_DPD_BG (bit 0 to 12) and E_DPD_VTTGEN (bit 16 to 27)

18.11.2.153 EMC_PMC_SCRATCH2_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH2:[PMC] Scratch register to allow storage of PMC controls for warmboot

18.11.2.154 EMC_PMC_SCRATCH3_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH3:[PMC] Scratch register to allow storage of PMC controls for warm boot

18.11.2.155 EMC_AUTO_CAL_CONFIG2_0

Auto-Calibration Master Compute Control Register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x458 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27:26	VREF0	AUTO_CAL_DQS_PD_TERM_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
25:24	VREF0	AUTO_CAL_DQS_PU_TERM_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
23:22	VREF0	AUTO_CAL_DQ_PD_TERM_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
21:20	VREF0	AUTO_CAL_DQ_PU_TERM_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
19:18	VREF0	AUTO_CAL_DQS_PD_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
17:16	VREF0	AUTO_CAL_DQS_PU_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
15:14	VREF0	AUTO_CAL_DQ_PD_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
13:12	VREF0	AUTO_CAL_DQ_PU_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
11:10	VREF0	AUTO_CAL_CMD_PD_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
9:8	VREF0	AUTO_CAL_CMD_PU_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
7:6	VREF0	AUTO_CAL_CA_PD_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2

Bit	Reset	Description
5:4	VREF0	AUTO_CAL_CA_PU_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
3:2	VREF0	AUTO_CAL_CLK_PD_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2
1:0	VREF0	AUTO_CAL_CLK_PU_CODE_SEL: [PMC] Select which Autocal vref measure cycle's results to use for the compute cycle 0 = VREF0 1 = VREF1 2 = VREF2

18.11.2.156 EMC_AUTO_CAL_CONFIG3_0

Auto-Calibration Master Compute Control Register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x45c | Read/Write: R/W | Reset: 0x00770000 (0bxxxxxxxxx111x111xx000000xx000000)

Bit	Reset	Description
22:20	0x7	AUTO_CAL_CLK_PD_SLOPE: [PMC] slope for CK pull-down value - range is 0.125 to 1.000 in steps of 0.125
18:16	0x7	AUTO_CAL_CLK_PU_SLOPE: [PMC] slope for CK pull-up value - range is 0.125 to 1.000 in steps of 0.125
13:8	0x0	AUTO_CAL_CLK_PD_OFFSET: [PMC] 2's complement offset for CK pull-down value
5:0	0x0	AUTO_CAL_CLK_PU_OFFSET: [PMC] 2's complement offset for CK pull-up value

18.11.2.157 EMC_TR_DVFS_0

Offset: 0x460 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	TRAINING_DVFS: Set for a DVFS that enables training 0 = DISABLED 1 = ENABLED

18.11.2.158 EMC_AUTO_CAL_CHANNEL_0

Autocal slave control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x464 | Read/Write: R/W | Reset: 0xc280030a (0b110x001010000000xxxx0011xx001010)

Bit	R/W	Reset	Description
31	RW	ENABLE	AUTO_CAL_UPDATE_IDLE: [PMC] Allow Autocal update outside of refresh if dramc is idle 0 = DISABLE 1 = ENABLE
30	RW	ENABLE	AUTO_CAL_STALL_ALL_TRAFFIC: [PMC] Stall all traffic for an Autocal update. When set to 0, will only stall RW traffic 0 = DISABLE 1 = ENABLE
29	RO	DISABLE	AUTO_CAL_COMPUTE_START_DVFS: [PMC] Writing a one to this bit triggers the Autocal compute FSM after DVFS shadow update 0 = DISABLE 1 = ENABLE
27:21	RW	0x14	CAL_WAIT_AFTER_DVFS: [PMC] Number of EMC clocks to wait after DVFS shadow update before resuming traffic to DRAM Note that this delay is applied serially after Autocal compute + transfer + update delay during DVFS (if enabled via AUTO_CAL_COMPUTE_START_DVFS trigger). For Autocal, this delay is meant to cover E_CAL_UPDATE pulse generation in pad macros for CK pads. This delay counter is used by DDLLCAL compute update also, and may need to be set to a higher value if AUTO_CAL_COMPUTE_START_DVFS is not triggered
20:16	RW	0x0	AUTO_CAL_UPDATE_TIMEOUT: [PMC] Timeout before forcing dramc idle to issue an Autocal update. A value of 0 disables the timeout This is a power of two value so a setting of TIMEOUT=8 will wait (1<<8) emcclk's
11:8	RW	0x3	AUTO_CAL_WAIT_BEFORE_UPDATE: [PMC] Number of EMC clocks to wait before issuing the Autocal update after the refresh/idle event. Minimum valid value is 0x3
5:0	RW	0xa	AUTO_CAL_NUM_STALL_CYCLES: [PMC] Number of EMC clocks to stall the DRAM bus for whenever Autocal codes are being updated

18.11.2.159 EMC_IBDLY_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x468 | Read/Write: R/W | Reset: 0x00000001 (0bxx00xxxxxxxxxxxxxxxxxxxx0000001)

Bit	Reset	Description
29:28	0x0	IBDLY_MODE: [PMC] OBDLY is enforced before write command is issued 0 = DISABLED 1 = AFTER_CMD 2 = RESERVED 3 = B4_CMD
6:0	0x1	IBDLY: [PMC] tells the chip when to change IBDLY for DQ/DQS.

18.11.2.160 EMC_OBDLY_0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000003 (0bxx00xxxxxxxxxxxxxxxxxxxx000011)

Bit	Reset	Description
29:28	0x0	OBDLY_MODE: [PMC] OBDLY is enforced before write command is issued 0 = DISABLED 1 = AFTER_CMD 2 = RESERVED 3 = B4_CMD
5:0	0x3	OBDLY: [PMC] tells the chip when to update OB trim values with respect to WDV

18.11.2.161 EMC_TXDSRVTTGEN_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x480 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	TXDSRVTTGEN: [PMC] Cycles to wait from DSR exit (VTTGEN drive normal), to when a DRAM transaction is allow to start.

18.11.2.162 EMC_WE_DURATION_0

DRAM timing parameter

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x48c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	WE_DURATION: [PMC] the number of cycles to assert write enable

18.11.2.163 EMC_WS_DURATION_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x490 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	WS_DURATION: [PMC] the number of cycles to assert write strobe

18.11.2.164 EMC_WEV_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x494 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	WEV: [PMC] the number of cycles to post (delay) write enable from being asserted to the RAMs. 40 = maximum

18.11.2.165 EMC_WSV_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x498 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	WSV: [PMC] the number of cycles to post (delay) write strobe from being asserted to the RAMs. \ 40 = maximum

18.11.2.166 EMC_CFG_3_0

EMC Configuration

Boot requirements:

- This register (except for fields DRAMC_PRE_B4_ACT, MRR_BYTESEL_X16, MRR_BYTESEL) should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- If the OS needs the MRR_BYTESEL* fields set to non-default values to perform a mode-register read, it needs to correctly program these values before performing the MRR.

Offset: 0x49c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	MRR_BYTESEL_X16: [PMC] If using 2 x16 DRAM on a single CS to form 32-bit wide data, indicates to which byte lane the second DRAM's byte 0 is connected.
2:0	0x0	MRR_BYTESEL: [PMC] Indicates which AP byte lane is connected to DRAM byte 0 (over which MRR data is returned).

18.11.2.167 EMC_MRW5_0

Command trigger: MRW5

Mode Register Write: LPDDRx-only version of MRS/EMRS

Offset: 0x4a0 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW5_DEV_SELECTN: active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW5_EXT_CNT: {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW5_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW5_MA: register address
7:0	0x0	MRW5_OP: data to be written

18.11.2.168 EMC_MRW6_0

Command trigger: MRW6

The following Mode Registers (MRW6-MRW15) are added for LPDDR4 support; however, they can be used for LPDDRx. They add sub-partition support. They always use short MRS wait count to satisfy tMRW.

Boot requirements:

- This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW6_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices 0x1 for dev1 0x2 to for only dev0,
27	0x0	USE_MRW6_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW6_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW6_SP_SELECTN: [PMC] active-low subp chip-select, both subp can be written at the same 0x2 writes OP_SP0 to subp0, 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW6_MA: [PMC] register address
15:8	0x0	MRW6_OP_SP1: [PMC] data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW6_OP_SP0: [PMC] data to be written to sub-partition 0 (subp0)

18.11.2.169 EMC_MRW7_0

Command trigger: MRW7

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW7_DEV_SELECTN: active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW7_EXT_CNT: {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW7_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW7_SP_SELECTN: active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW7_MA: register address
15:8	0x0	MRW7_OP_SP1: data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW7_OP_SP0: data to be written to sub-partition 0 (subp0)

18.11.2.170 EMC_MRW8_0

Command trigger: MRW8

Offset: 0x4ac | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW8_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW8_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW8_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW8_SP_SELECTN: [PMC] active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1

Bit	Reset	Description
23:16	0x0	MRW8_MA: [PMC] register address
15:8	0x0	MRW8_OP_SP1: [PMC] data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW8_OP_SP0: [PMC] data to be written to sub-partition 0 (subp0)

18.11.2.171 EMC_MRW9_0

Command trigger: MRW9

Offset: 0x4b0 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW9_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW9_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW9_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW9_SP_SELECTN: [PMC] active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW9_MA: [PMC] register address
15:8	0x0	MRW9_OP_SP1: [PMC] data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW9_OP_SP0: [PMC] data to be written to sub-partition 0 (subp0)

18.11.2.172 EMC_MRW10_0

Command trigger: MRW10

Offset: 0x4b4 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW10_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW10_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW10_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW10_SP_SELECTN: [PMC] active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW10_MA: [PMC] register address
15:8	0x0	MRW10_OP_SP1: [PMC] data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW10_OP_SP0: [PMC] data to be written to sub-partition 0 (subp0)

18.11.2.173 EMC_MRW11_0

Command trigger: MRW11

Offset: 0x4b8 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW11_DEV_SELECTN: active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.

Bit	Reset	Description
27	0x0	USE_MRW11_EXT_CNT: {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW11_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW11_SP_SELECTN: active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW11_MA: register address
15:8	0x0	MRW11_OP_SP1: data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW11_OP_SP0: data to be written to sub-partition 0 (subp0)

18.11.2.174 EMC_MRW12_0

Command trigger: MRW12

Offset: 0x4bc | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW12_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW12_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW12_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW12_SP_SELECTN: [PMC] active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW12_MA: [PMC] register address
15:8	0x0	MRW12_OP_SP1: [PMC] data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW12_OP_SP0: [PMC] data to be written to sub-partition 0 (subp0)

18.11.2.175 EMC_MRW13_0

Command trigger: MRW13

Offset: 0x4c0 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW13_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW13_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW13_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW13_SP_SELECTN: [PMC] active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW13_MA: [PMC] register address
15:8	0x0	MRW13_OP_SP1: [PMC] data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW13_OP_SP0: [PMC] data to be written to sub-partition 0 (subp0)

18.11.2.176 EMC_MRW14_0

Command trigger: MRW14

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW14_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW14_EXT_CNT: [PMC] {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW14_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW14_SP_SELECTN: [PMC] active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW14_MA: [PMC] register address
15:8	0x0	MRW14_OP_SP1: [PMC] data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW14_OP_SP0: [PMC] data to be written to sub-partition 0 (subp0)

18.11.2.177 EMC_MRW15_0

Command trigger: MRW15

Offset: 0x4d0 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00000000000000000000000000)

Bit	Reset	Description
31:30	0x0	MRW15_DEV_SELECTN: active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW15_EXT_CNT: {USE_MRS_EXT_CNT, USE_MRS_LONG_CNT}: 00 = SHORT, 01 = LONG, 10 = EXT1, 11=EXT2
26	SHORT	USE_MRW15_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
25:24	0x0	MRW15_SP_SELECTN: active-low subp chip-select, both subp can be written at the same time 0x2 writes OP_SP0 to subp0 0x1 writes OP_SP1 to subp1 0x0 writes OP_SP0 to subp0 and OP_SP1 to subp1
23:16	0x0	MRW15_MA: register address
15:8	0x0	MRW15_OP_SP1: data to be written to sub-partition 1 (subp1)
7:0	0x0	MRW15_OP_SP0: data to be written to sub-partition 0 (subp0)

18.11.2.178 EMC_CFG_SYNC_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CHANNEL_SYNC: When this register is accessed on one channel all subsequent priv accesses to that channel are blocked till the other channel sees an access to the same register - this is to get both channels in sync IMPORTANT: This should not be written in single channel operation

18.11.2.179 EMC_FDPD_CTRL_CMD_NO_RAMP_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CMD_DPD_NO_RAMP_ENABLE: [PMC] FDPD on CMD brick through TX_EN - no ramping for di/dt, disabled before DVFS and re-enabled after 0 = DISABLED 1 = ENABLED

18.11.2.180 EMC_WDV_CHK_0

DRAMC timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x4e0 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000110)

Bit	Reset	Description
5:0	0x6	WDV_CHK_BASE: [PMC] This field, in conjunction with CFG_2.DRAMC_WD_CHK_POLICY, establishes DRAMC write data ready handling. This field is in terms of EMC clocks.

18.11.2.181 EMC_CFG_PIPE_2_0

pipe Configuration Register

This pipe stage is present inside the pad macro not at the top level.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x554 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK11: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick11
26	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK10: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick10
25	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK9: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick9
24	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK8: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick8
23	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK7: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick7
22	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK6: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick6
21	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK5: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick5
20	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK4: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick4
19	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK3: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick3
18	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK2: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick2
17	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK1: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick1
16	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE3_BRICK0: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for cmd pin of brick0
11	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK11: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick11
10	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK10: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick10

Bit	Reset	Description
9	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK9: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick9
8	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK8: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick8
7	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK7: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick7
6	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK6: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick6
5	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK5: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick5
4	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK4: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick4
3	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK3: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick3
2	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK2: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick2
1	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK1: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick1
0	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE3_BRICK0: [PMC] 1 = bypass pipeline stage3 between EMC and pad-macro for brick0

18.11.2.182 EMC_CFG_PIPE_CLK_0

Offset: 0x558 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	CLK_ALWAYS_ON	PIPE_CLK_ENABLE_OVERRIDE: [PMC] 0 = CLK_GATED 1 = CLK_ALWAYS_ON 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

18.11.2.183 EMC_CFG_PIPE_1_0

pipe Configuration Register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x55c | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK11: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick11
26	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK10: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick10
25	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK9: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick9
24	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK8: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick8
23	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK7: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick7
22	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK6: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick6
21	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK5: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick5
20	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK4: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick4

Bit	Reset	Description
19	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK3: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick3
18	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK2: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick2
17	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK1: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick1
16	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE2_BRICK0: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for cmd pin of brick0
11	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK11: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick11
10	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK10: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick10
9	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK9: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick9
8	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK8: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick8
7	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK7: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick7
6	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK6: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick6
5	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK5: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick5
4	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK4: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick4
3	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK3: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick3
2	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK2: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick2
1	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK1: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick1
0	0x0	EMC2PMACRO_CFG_BYPASS_CMD_OB_PIPE1_BRICK0: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for cmd pin of brick0

18.11.2.184 EMC_CFG_PIPE_0

Pipe Configuration Register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x560 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK11: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick11
26	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK10: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick10
25	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK9: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick9
24	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK8: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick8
23	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK7: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick7
22	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK6: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick6
21	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK5: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick5
20	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK4: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick4

19	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK3: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick3
18	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK2: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick2
17	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK1: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick1
16	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE2_BRICK0: [PMC] 1 = bypass pipeline stage2 between EMC and pad-macro for brick0
11	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK11: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick11
10	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK10: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick10
9	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK9: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick9
8	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK8: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick8
7	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK7: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick7
6	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK6: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick6
5	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK5: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick5
4	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK4: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick4
3	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK3: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick3
2	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK2: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick2
1	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK1: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick1
0	0x0	EMC2PMACRO_CFG_BYPASS_OB_PIPE1_BRICK0: [PMC] 1 = bypass pipeline stage1 between EMC and pad-macro for brick0

18.11.2.185 EMC_QPOP_0

DRAM timing parameter

This register is shadowed

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the
- Boot ROM during warm boot.

Offset: 0x564 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxx0000000xxxxxxxxx0000110)

Bit	Reset	Description
22:16	0x0	QPOP_PREAMBLE: [PMC] time from read command to pop preamble from the pad macro FIFO. EMC ignores this field if FBIO_CFG7.QPOP_RD_PREAMBLE_TOGGLE=0. 60 = MAX_1TO1 120 = MAX_2TO1
6:0	0x6	QPOP: [PMC] time from read command to pop data from the pad macro FIFO. 60 = MAX_1TO1 120 = MAX_2TO1

18.11.2.186 EMC_QUSE_WIDTH_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x568 | Read/Write: R/W | Reset: 0x00000003 (0bxx00xxxxxxxxxxxxxxxxxxxxxxxx0011)

Bit	Reset	Description
29	0x0	QUSE_SHORTEN_2UI: [PMC] Shorten QUSE 2 UI (1 DRAM clock)
28	0x0	QUSE_EXTEND_UI: [PMC] Extend QUSE 1 UI (1/2 clock)
3:0	0x3	QUSE_DURATION: [PMC] QUSE_DURATION. minimum value = 1 (ddr3/lp3), 2 (LP4), results 8/16 tCLK quse window

18.11.2.187 EMC_PUTERM_WIDTH_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Set both RXTERM_DURATION and CFG_STATIC_TERM to disable terminations.

RXTERM defines start of term window. RX_TERM_MODE controls termination type

Offset: 0x56c | Read/Write: R/W | Reset: 0x00000005 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxx0101)

Bit	Reset	Description
31	0x0	CFG_STATIC_TERM: [PMC] IOBRICK internally disables termination based on DOE. Core sends static '1' on term_enable
3:0	0x5	RXTERM_DURATION: [PMC] Additional PUTERM duration apart from default BL/2 -- set to 0 to disable termination

18.11.2.188 EMC_BGBIAS_CTL0_0

BGBIAS Pad controls

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x570 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	BIAS0_DSC_E_PWRD_IBIAS_RX: [PMC] Bias current going out to brick receiver
2	0x0	BIAS0_DSC_E_PWRD_IBIAS_VTTGEN: [PMC] Active High. Disables biasing current going out to VTTGEN cells
1	0x0	BIAS0_DSC_E_PWRD: [PMC] Active High. Disables all DC current paths within entire cell.
0	0x0	BIAS0_DSC_BG_SEL_N: [PMC] Active low; 0= select BandGap reference current;

18.11.2.189 EMC_AUTO_CAL_CONFIG7_0

Autocal master compute control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x574 | Read/Write: R/W | Reset: 0x00770000 (0bxxxxxxxxx111x111xx000000xx000000)

Bit	Reset	Description
22:20	0x7	AUTO_CAL_DQS_PD_SLOPE: [PMC] slope for DQS pull-down value - range is 0.125 to 1.000 in steps of 0.125
18:16	0x7	AUTO_CAL_DQS_PU_SLOPE: [PMC] slope for DQS pull-up value - range is 0.125 to 1.000 in steps of 0.125
13:8	0x0	AUTO_CAL_DQS_PD_OFFSET: [PMC] 2's complement offset for DQS pull-down value
5:0	0x0	AUTO_CAL_DQS_PU_OFFSET: [PMC] 2's complement offset for DQS pull-up value

18.11.2.190 EMC_XM2COMPPADCTRL2_0

Autocal COMP pad control register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x578 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
17	DISABLE	EMC2TMC_CFG_XM2COMP_PU_E_WKPD: [PMC] Enable weak pull-down 0 = DISABLE 1 = ENABLE
16	DISABLE	EMC2TMC_CFG_XM2COMP_PU_E_WKPU: [PMC] Enable weak pull-up 0 = DISABLE 1 = ENABLE
15:14	0x0	EMC2TMC_CFG_XM2COMP_PU_DRVDN_ZCTRL: [PMC] Drive pull-down impedance calibration select. 0x0 = 40ohms, 0x1 = 60ohms, 0x2 = 80ohms, 0x3 = 120ohms
13:12	0x0	EMC2TMC_CFG_XM2COMP_PU_DRVUP_ZCTRL: [PMC] Drive pull-up impedance calibration select. 0x0 = 40ohms, 0x1 = 60ohms, 0x2 = 80ohms, 0x3 = 120ohms
11:10	0x0	EMC2TMC_CFG_XM2COMP_PU_VDDA_MODE: [PMC] Function regulator rail from adjacent IOBRICK. 11 to enable local switch to have better power delivery
9:8	0x0	EMC2TMC_CFG_XM2COMP_PU_VAUXP_MODE: [PMC] Function regulator rail from adjacent IOBRICK. 11 to enable local switch to have better power delivery
7:6	0x0	EMC2TMC_CFG_XM2COMP_PU_VCLAMP_MODE: [PMC] Function regulator rail from adjacent IOBRICK. 11 to enable local switch to have better power delivery
5:2	0x0	EMC2TMC_CFG_XM2COMP_PU_TEST_MODE: [PMC] Test mode select
1:0	SCHMITT	EMC2TMC_CFG_XM2COMP_PU_RX_MODE: [PMC] Set front-end RX type 0 = SCHMITT 1 = RFU0 2 = DIFFAMP 3 = RFU1

18.11.2.191 EMC_COMP_PAD_SW_CTRL_0

Autocal COMP pad register for Software control of pad inputs

Offset: 0x57c | Read/Write: R/W | Reset: 0x738000f0 (0bx1110x11100000000000000x011110000)

Bit	Reset	Description
30	ENABLE	AUTO_CAL_SW_CTRL_PD_CAL_IVREF_CAL: Software control for IVREF_CAL_MODE pin of COMP pad slice used for pull-down calibration 0 = DISABLE 1 = ENABLE
29	ENABLE	AUTO_CAL_SW_CTRL_PD_CAL_E_IVREF: Software control for E_IVREF pin of COMP pad slice used for pull-down calibration 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	ENABLE	AUTO_CAL_SW_CTRL_PD_CAL_E_INPUT: Software control for E_INPUT pin of COMP pad slice used for pull-down calibration 0 = DISABLE 1 = ENABLE
27	ENABLE	AUTO_CAL_SW_CTRL_PD_CAL_EN: Software control for EN pin of COMP pad slice used for pull-down calibration 0 = ENABLE 1 = DISABLE
25	ENABLE	AUTO_CAL_SW_CTRL_PU_CAL_IVREF_CAL: Software control for IVREF_CAL_MODE pin of COMP pad slice used for pull-up calibration 0 = DISABLE 1 = ENABLE
24	ENABLE	AUTO_CAL_SW_CTRL_PU_CAL_E_IVREF: Software control for E_IVREF pin of COMP pad slice used for pull-up calibration 0 = DISABLE 1 = ENABLE
23	ENABLE	AUTO_CAL_SW_CTRL_PU_CAL_E_INPUT: Software control for E_INPUT pin of COMP pad slice used for pull-up calibration 0 = DISABLE 1 = ENABLE
22	ENABLE	AUTO_CAL_SW_CTRL_PU_CAL_EN: Software control for TX_EN pin of COMP pad slice used for pull-up calibration 0 = ENABLE 1 = DISABLE
21:16	0x0	AUTO_CAL_SW_CTRL_DRVDN: Software control for TX_DRVDN pin of COMP pad slices used for pull-up and pull-down calibration
15:10	0x0	AUTO_CAL_SW_CTRL_DRVUP: Software control for TX_DRVUP pin of COMP pad slices used for pull-up and pull-down calibration
8:2	0x3c	AUTO_CAL_SW_CTRL_IVREF_LVL: Software control for IVREF_LVL, shared between COMP pad slices used for pull-up and pull-down calibration
1	DISABLE	AUTO_CAL_SW_CTRL_BG_E_PWRD: Software control for BG_E_PWRD 0 = DISABLE 1 = ENABLE
0	DISABLE	AUTO_CAL_SW_CTRL: Enable Software control of COMP pad inputs. Make sure AUTO_CAL_ENABLE is set to DISABLED first 0 = DISABLE 1 = ENABLE

18.11.2.192 EMC_REFCTRL2_0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x580 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxx000xxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31	DISABLED	REFPB_VALID: [PMC] Specifies if REFpb operation is enabled. Not valid unless REFCTRL_0_REF_VALID is ENABLED 0 = DISABLED 1 = ENABLED
26:24	0x0	REFPB_PD_THRESHOLD: [PMC] When in PD, issue REFpb if pending REFpb % num_banks is less than threshold. otherwise switch to REFab in PD A value of 0x0 will always issue REFab in PD
0	DISABLED	REFRESH_PER_DEVICE: [PMC] Controls whether refresh commands are sent independantly for each device, or are shared. 0 = DISABLED 1 = ENABLED

18.11.2.193 EMC_FBIO_CFG7_0

FBIO configuration Register

Offset: 0x584 | Read/Write: R/W | Reset: 0x0000002a (0b00000000000000000000000000101010)

Bit	Reset	Description
31:17	0x0	TEMP_REG: Reserved.
16	0x0	FORCE_CMD_PHASE_EQ0: [PMC] Setting this bit forces DRAM cmds to always be launched on phase 1. Bit is ignored if (DRAM_EMCCLK_2TO1 == ONEX).
15	0x0	DISABLE_CCDP1_WR_SPACING: [PMC] Setting this bit prevents write command spacing of tCCD+1. Allowed write spacing will then be tCCD or tCCD > 1.
14	0x0	DISABLE_CCDP1_RD_SPACING: [PMC] Setting this bit prevents read command spacing of tCCD+1. Allowed read spacing will then be tCCD or tCCD > 1. Set if using QPOP_RD_PREAMBLE_TOGGLE
13	X32	SUBP_ADDR_MODE: [PMC] Selects dram subp addr width. This affects mc column to dram column translation. Expected usage is X16 for LPDDR4 and X32 otherwise. 0 = X32 1 = X16
12	DISABLED	ZQCAL_LATCH_ENABLE: [PMC] Enables EMC to send an LPDDR4 MPC ZQCal Latch command as part of ZQ Calibration process. 0 = DISABLED 1 = ENABLED
11	DISABLED	ZQCAL_MPC_ENABLE: [PMC] Enables EMC to send ZQCal command as LPDDR4 MPC ZQCal Start command 0 = DISABLED 1 = ENABLED
10	DISABLED	QPOP_RD_PREAMBLE_TOGGLE: [PMC] enable qpop filter for LPDDR4 read preamble toggle 0 = DISABLED 1 = ENABLED
9	ACTIVE_LOW	CS_POLARITY: [PMC] Determines polarity of DRAM chip select (CS). LPDDR4 implements an active high CS. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
8	DISABLED	MRR_DBI: [PMC] If RD_DBI=ENABLED, enables EMC to apply data bus inversion protocol to in-coming mode register read data (LPDDR4). EMC ignores this bit otherwise. This allows EMC to support DRAM which does not apply dbi to MRR data. 0 = DISABLED 1 = ENABLED
7	DISABLED	RD_DBI: [PMC] Enables EMC to apply data bus inversion protocol to in-coming read data (LPDDR4). If read dbi is enabled in DRAM, this bit must also be enabled. 0 = DISABLED 1 = ENABLED
6	DISABLED	WR_DBI: [PMC] Enables EMC to use write data bus inversion protocol (LPDDR4). If enabled, EMC will apply data bus inversion to any valid bytes which have more than 4 bits set. If this bit is enabled, write dbi must also be enabled in DRAM. 0 = DISABLED 1 = ENABLED
5	ENABLED	MASKED_WR: [PMC] Enables EMC to issue LPDDR4 masked write command. 0 = DISABLED 1 = ENABLED
4	DISABLED	LPDDR4_CMD_MAP: [PMC] Enables EMC to use LPDDR4 command mapping for DRAM commands. 0 = DISABLED 1 = ENABLED
3	ENABLED	QUSE_CCDP1_EXTEND: [PMC] Enables extension of QUSE through tCCD+1 spacing. Required for direct QUSE. Must be disabled for RPRE toggle 0 = DISABLED 1 = ENABLED
2	DISABLE	CH1_ENABLE: [PMC] 0 = DISABLE 1 = ENABLE
1	ENABLE	CH0_ENABLE: [PMC] 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	ONEX	DRAM_EMCCLK_2TO1: [PMC] Set to enabled if dramclk is 2x faster than emcclk 0 = ONEX 1 = TWOX

18.11.2.194 EMC_DATA_BRLSHFT_0_0

FBIO configuration Register

DQ/DQS BRLSHFTS

Rank 0 Per byte OB barrleshift setting

Offset: 0x588 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:21	0x0	RANK0_BYTE7_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
20:18	0x0	RANK0_BYTE6_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
17:15	0x0	RANK0_BYTE5_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
14:12	0x0	RANK0_BYTE4_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
11:9	0x0	RANK0_BYTE3_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
8:6	0x0	RANK0_BYTE2_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
5:3	0x0	RANK0_BYTE1_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
2:0	0x0	RANK0_BYTE0_DATA_BRLSHFT: [PMC] Per byte barrelshift setting

18.11.2.195 EMC_DATA_BRLSHFT_1_0

FBIO Configuration Register

Rank1 Per byte OB barrleshift setting

Offset: 0x58c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:21	0x0	RANK1_BYTE7_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
20:18	0x0	RANK1_BYTE6_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
17:15	0x0	RANK1_BYTE5_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
14:12	0x0	RANK1_BYTE4_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
11:9	0x0	RANK1_BYTE3_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
8:6	0x0	RANK1_BYTE2_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
5:3	0x0	RANK1_BYTE1_DATA_BRLSHFT: [PMC] Per byte barrelshift setting
2:0	0x0	RANK1_BYTE0_DATA_BRLSHFT: [PMC] Per byte barrelshift setting

18.11.2.196 EMC_RFCPB_0

DRAM timing parameter

Offset: 0x590 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxx000111111)

Bit	Reset	Description
8:0	0x3f	RFCPB: [PMC] specifies the per-bank auto refresh cycle time. This is the minimum number of cycles between a per-bank auto refresh command and a subsequent per-bank refresh (any bank) or activate command to that bank.

18.11.2.197 EMC_DQS_BRLSHFT_0_0

FBIO Configuration Register

Rank 0 Per byte OB barrelshift setting

Offset: 0x594 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:21	0x0	RANK0_BYTE7_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
20:18	0x0	RANK0_BYTE6_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
17:15	0x0	RANK0_BYTE5_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
14:12	0x0	RANK0_BYTE4_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
11:9	0x0	RANK0_BYTE3_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
8:6	0x0	RANK0_BYTE2_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
5:3	0x0	RANK0_BYTE1_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
2:0	0x0	RANK0_BYTE0_DQS_BRLSHFT: [PMC] Per byte barrelshift setting

18.11.2.198 EMC_DQS_BRLSHFT_1_0

FBIO Configuration Register

Rank1 Per byte OB barrelshift setting

Offset: 0x598 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:21	0x0	RANK1_BYTE7_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
20:18	0x0	RANK1_BYTE6_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
17:15	0x0	RANK1_BYTE5_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
14:12	0x0	RANK1_BYTE4_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
11:9	0x0	RANK1_BYTE3_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
8:6	0x0	RANK1_BYTE2_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
5:3	0x0	RANK1_BYTE1_DQS_BRLSHFT: [PMC] Per byte barrelshift setting
2:0	0x0	RANK1_BYTE0_DQS_BRLSHFT: [PMC] Per byte barrelshift setting

18.11.2.199 EMC_CMD_BRLSHFT_0_0

CMD/ADR BRLSHFTS

In LP4 mode SUBP0 address bits will be on one brick and SUBP1 address on another; so different 1T settings will be needed for different long trimmers.

In legacy (LP3) modes, the same 1T setting will need to be programmed for SUBP0 and SUBP1 because addresses belonging to the same subpartition can end up on different bricks with different long trimmer settings.

In summary, in LP4 - Training will set SUBP0 1T and SUBP1 1T; only software will set 2T for CKE/RST.

In Legacy modes - Software can only set both SUBP0 1T and SUBP1 1T to the same value. Also, software can set SUBP0 2T and SUBP1 2T to the same value.

1T LP4 ADR barrelshift setting on Ch0

These will be set by the training engine in LP4 mode. In Legacy modes, these will be set by software (if needed)

Offset: 0x59c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:3	0x0	CH0_SUBP1_1T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting

2:0	0x0	CH0_SUBP0_1T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting

18.11.2.200 EMC_CMD_BRLSHFT_1_0

1T LP4 ADR barrelshift setting on Ch1

These will be set by the training engine in LP4 mode. In Legacy modes, these will be set by software (if needed)

Offset: 0x5a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:3	0x0	CH1_SUBP1_1T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting
2:0	0x0	CH1_SUBP0_1T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting

18.11.2.201 EMC_CMD_BRLSHFT_2_0

2T signal barrelshift setting on Ch1

These will not be set by the training engine in LP4 mode and will be applied only to CKE and RST. In legacy modes, these will be applied to non-ADR signals like RAS, CAS, WE

Offset: 0x5a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:3	0x0	CH0_SUBP1_2T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting
2:0	0x0	CH0_SUBP0_2T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting

18.11.2.202 EMC_CMD_BRLSHFT_3_0

2T signal barrleshift setting on Ch1

These will not be set by the training engine in LP4 mode and will be applied only to CKE and RST. In legacy modes these will be applied to non-ADR signals like RAS, CAS, WE.

Offset: 0x5a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:3	0x0	CH1_SUBP1_2T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting
2:0	0x0	CH1_SUBP0_2T_BRLSHFT: [PMC] Per Subp CMD/ADR barrelshift setting

18.11.2.203 EMC_QUSE_BRLSHFT_0_0

QUSE BRLSHFTS

Per byte OB barrelshift setting

Offset: 0x5ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:15	0x0	RANK0_BYTE3_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
14:10	0x0	RANK0_BYTE2_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
9:5	0x0	RANK0_BYTE1_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
4:0	0x0	RANK0_BYTE0_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting

18.11.2.204 EMC_AUTO_CAL_CONFIG4_0

Autocal Master Compute Control Register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x5b0 | Read/Write: R/W | Reset: 0x00770000 (0bxxxxxxxxx111x111xx000000xx000000)

Bit	Reset	Description
22:20	0x7	AUTO_CAL_CA_PD_SLOPE: [PMC] slope for CA pull-down value - range is 0.125 to 1.000 in steps of 0.125
18:16	0x7	AUTO_CAL_CA_PU_SLOPE: [PMC] slope for CA pull-up value - range is 0.125 to 1.000 in steps of 0.125
13:8	0x0	AUTO_CAL_CA_PD_OFFSET: [PMC] 2's complement offset for CA pull-down value
5:0	0x0	AUTO_CAL_CA_PU_OFFSET: [PMC] 2's complement offset for CA pull-up value

18.11.2.205 EMC_AUTO_CAL_CONFIG5_0

Autocal master compute control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x5b4 | Read/Write: R/W | Reset: 0x00770000 (0bxxxxxxxxx111x111xx000000xx000000)

Bit	Reset	Description
22:20	0x7	AUTO_CAL_CMD_PD_SLOPE: [PMC] slope for CMD pull-down value - range is 0.125 to 1.000 in steps of 0.125
18:16	0x7	AUTO_CAL_CMD_PU_SLOPE: [PMC] slope for CMD pull-up value - range is 0.125 to 1.000 in steps of 0.125
13:8	0x0	AUTO_CAL_CMD_PD_OFFSET: [PMC] 2's complement offset for CMD pull-down value
5:0	0x0	AUTO_CAL_CMD_PU_OFFSET: [PMC] 2's complement offset for CMD pull-up value

18.11.2.206 EMC_QUSE_BRLSHFT_1_0

Per byte OB barrelshift setting

Offset: 0x5b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:15	0x0	RANK0_BYTE7_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
14:10	0x0	RANK0_BYTE6_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
9:5	0x0	RANK0_BYTE5_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
4:0	0x0	RANK0_BYTE4_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting

18.11.2.207 EMC_QUSE_BRLSHFT_2_0

Per byte OB barrelshift setting

Offset: 0x5bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:15	0x0	RANK1_BYTE3_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
14:10	0x0	RANK1_BYTE2_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
9:5	0x0	RANK1_BYTE1_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
4:0	0x0	RANK1_BYTE0_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting

18.11.2.208 EMC_CCDMW_0

DRAM timing parameter

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x5c0 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	CCDMW: [PMC] This field supports LPDDR4 DRAMs. It specifies the DRAM CAS to CAS delay timing when the current command is any type of write and the next command is a masked write to the same bank. The EMC ignores this field if FBIO_CFG7.MASKED_WR is disabled.

18.11.2.209 EMC_QUSE_BRLSHFT_3_0

Per byte OB barrelshift setting

Offset: 0x5c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:15	0x0	RANK1_BYTE7_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
14:10	0x0	RANK1_BYTE6_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
9:5	0x0	RANK1_BYTE5_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting
4:0	0x0	RANK1_BYTE4_QUSE_BRLSHFT: [PMC] Per byte quse barrelshift setting

18.11.2.210 EMC_FBIO_CFG8_0

FBIO configuration Register

The QRST,QUSE,RDV fields specify the delays from read to internal timing signals. These fields should be set as follows:

QUSE = CAS_LATENCY - 1 non-mobile SDRAM

Offset: 0x5c8 | Read/Write: R/W | Reset: 0x0fff0000 (0bxxxx1111111111110xxxxxxxxxxxxxxx)

Bit	Reset	Description
27	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK11: [PMC] 1= enables the clock to data control signals in pipestages of brick11. set to =0 in cmd bricks
26	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK10: [PMC] 1= enables the clock to data control signals in pipestages of brick10. set to =0 in cmd bricks
25	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK9: [PMC] 1= enables the clock to data control signals in pipestages of brick9. set to =0 in cmd bricks
24	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK8: [PMC] 1= enables the clock to data control signals in pipestages of brick8. set to =0 in cmd bricks
23	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK7: [PMC] 1= enables the clock to data control signals in pipestages of brick7. set to =0 in cmd bricks
22	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK6: [PMC] 1= enables the clock to data control signals in pipestages of brick6. set to =0 in cmd bricks
21	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK5: [PMC] 1= enables the clock to data control signals in pipestages of brick5. set to =0 in cmd bricks
20	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK4: [PMC] 1= enables the clock to data control signals in pipestages of brick4. set to =0 in cmd bricks
19	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK3: [PMC] 1= enables the clock to data control signals in pipestages of brick3. set to =0 in cmd bricks
18	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK2: [PMC] 1= enables the clock to data control signals in pipestages of brick2. set to =0 in cmd bricks

Bit	Reset	Description
17	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK1: [PMC] 1= enables the clock to data control signals in pipestages of brick1. set to =0 in cmd bricks
16	0x1	EMC2PMACRO_CFG_PIPE_DATA_MODE_BRICK0: [PMC] 1= enables the clock to data control signals in pipestages of brick0. set to =0 in cmd bricks
15	0x0	Reserved. Write zero.

18.11.2.211 EMC_AUTO_CAL_CONFIG6_0

Autocal master compute control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x5cc | Read/Write: R/W | Reset: 0x00770000 (0bxxxxxxxxxx111x111xx000000xx000000)

Bit	Reset	Description
22:20	0x7	AUTO_CAL_DQ_PD_SLOPE: [PMC] slope for DQ pull-down value - range is 0.125 to 1.000 in steps of 0.125
18:16	0x7	AUTO_CAL_DQ_PU_SLOPE: [PMC] slope for DQ pull-up value - range is 0.125 to 1.000 in steps of 0.125
13:8	0x0	AUTO_CAL_DQ_PD_OFFSET: [PMC] 2's complement offset for DQ pull-down value
5:0	0x0	AUTO_CAL_DQ_PU_OFFSET: [PMC] 2's complement offset for DQ pull-up value

18.11.2.212 EMC_PROTOBIST_CONFIG_ADR_1_0

Protocol BIST register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x5d0 | Read/Write: WO | Reset: 0x30000000 (0bxx11000000000000000000000000000)

Bit	Reset	Description
29	0x1	PROTOBIST_APC: Controls the auto-precharge function of transaction.
28	0x1	PROTOBIST_A_RDY: Indicates Activate interface is ready.
27:12	0x0	PROTOBIST_A_ROW: ROW control during protobist mode.
11	0x0	PROTOBIST_A_REF: REFRESH control during Activate request in protobist
10:9	0x0	PROTOBIST_A_DEV: DEVSEL control during Activate request in protobist
8:6	0x0	PROTOBIST_BANK: BANK control during transfer request in protobist
5:3	0x0	PROTOBIST_A_BANK: BANK control during Activate request in protobist
2:0	0x0	PROTOBIST_A_BANK1: BANK1 control during Activate request in protobist

18.11.2.213 EMC_PROTOBIST_CONFIG_ADR_2_0

Protocol BIST register

Offset: 0x5d4 | Read/Write: WO | Reset: 0x08000101 (0bxxx01000000000000000000100000001)

Bit	Reset	Description
28	0x0	PROTOBIST_SWAP: Swap control during protobist mode
27:26	0x2	PROTOBIST_SIZE: Size control during protobist mode
25	0x0	PROTOBIST_MASKED_WRITE: Mask write control during protobist mode
24	0x0	PROTOBIST_T_DEV: DEV control during Xfre request of protobist mode

Bit	Reset	Description
23:21	0x0	PROTOBIST_T_COL1: Column1 control during transfer request during protobist
20:9	0x0	PROTOBIST_T_COL: COlumn control during protobist
8	0x1	PROTOBIST_T_RDY: Indicates transfer interface is ready.
7:0	0x1	PROTOBIST_CGID: Controls the request ID of transaction.

18.11.2.214 EMC_PROTOBIST_MISC_0

Protocol BIST register

Offset: 0x5d8 | Read/Write: R/W | Reset: 0x0000X000 (0bxxxxxxxxxxxxx000x000000000000000)

Bit	R/W	Reset	Description
18:16	WO	0x0	PROTOBIST_REQ
15	RO	X	PROTOBIST_READ_ACK
14	RW	0x0	PROTOBIST_START
13	WO	0x0	PROTOBIST_W_BE: Protobist mode control for byte enable.
12	WO	0x0	PROTOBIST_MODE: Protobist mode control, indicates bist mode.
11:8	WO	0x0	PROTOBIST_RDI_SEL: RDI select is MUX control for read data.
7:0	WO	0x0	PROTOBIST_TAG: Tag control during protobist mode

18.11.2.215 EMC_PROTOBIST_WDATA_LOWER_0

Protocol BIST register

Offset: 0x5dc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PROTOBIST_DATA_LOWER: 1st 32 bit Data for protobist mode

18.11.2.216 EMC_PROTOBIST_WDATA_UPPER_0

Protocol BIST register

Offset: 0x5e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PROTOBIST_DATA_UPPER: 2nd 32 bit Data for protobist mode

18.11.2.217 EMC_PROTOBIST_RDATA_0

Protocol BIST register

Offset: 0x5ec | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PROTOBIST_RDATA: read data after read Xaction during PBIST mode

18.11.2.218 EMC_DLL_CFG_0_0

DLL calibration configuration register 0

Offset: 0x5e4 | Read/Write: R/W | Reset: 0x01a340ff (0bxx000001101000110100000011111111)

Bit	Reset	Description
29	0x0	DDLLCAL_CTRL_IGNORE_START: [PMC] Ignore DLLCAL start trim priv value
28	0x0	DDLLCAL_CTRL_DUAL_PASS_LOCK: [PMC] 2 Pass Lock

Bit	Reset	Description
27:24	0x1	DDLLCAL_CTRL_STEP_SIZE: [PMC] Calibration step size
23:20	0xa	DDLLCAL_CTRL_END_COUNT: [PMC] End count
19:16	0x3	DDLLCAL_CTRL_FILTER_BITS: [PMC] Filter
15:12	0x4	DDLLCAL_CTRL_SAMPLE_COUNT: [PMC] No. of samples to take (power of 2)
11:4	0xf	DDLLCAL_CTRL_SAMPLE_DELAY: [PMC] delay to allow DLL value to settle
3:0	0xf	DDLLCAL_UPDATE_CNT_LIMIT: [PMC] Wait cycles for priv update

18.11.2.219 EMC_DLL_CFG_1_0

DLL calibration configuration register 1

Offset: 0x5e8 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxx00000000000100000)

Bit	Reset	Description
16	0x0	DDLL_BYPASS: [PMC] Bypass
15:12	0x0	DDLL_RFU: [PMC] RFU
11	0x0	E_DDLL_PWRD: [PMC] Master dLL powerdown
10:0	0x20	DDLLCAL_CTRL_START_TRIM: [PMC] Calibration start value

18.11.2.220 EMC_CONFIG_SAMPLE_DELAY_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Some fields in this register are shadowed: see usage note in *Section 18.11.2: EMC Registers*.

Offset: 0x5f0 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0100000)

Bit	Reset	Description
6:0	0x20	PMACRO_SAMPLE_DELAY: [PMC] Number of emcclk's from sending pad macro configuration read, to sampling of the read return data.

18.11.2.221 EMC_CFG_UPDATE_0

CFG_UPDATE is used to control special features of the timing updates.

Offset: 0x5f4 | Read/Write: R/W | Reset: 0x70000301 (0b0111xxxxxxxxxxxxxxxx011xxxxx001)

Bit	Reset	Description
31	DISABLED	STALL_READS_DURING_TIMING_UPDATE: [PMC] Disallow configuration reads during a timing update 0 = DISABLED 1 = ENABLED
30	ENABLED	STALL_WRITES_DURING_TIMING_UPDATE: [PMC] Disallow configuration writes during a timing update 0 = DISABLED 1 = ENABLED
29	ENABLED	LINKED_TIMING_UPDATE_TIMEOUT: [PMC] Set to ENABLED to have the linked timing update timeout after 16k emcclk's. When DISABLED the linked update can cause the EMC to hang if the timing update is not sent to both channels in dual channel operation 0 = DISABLED 1 = ENABLED
28	ENABLED	LINKED_TIMING_UPDATE: [PMC] Set to DISABLED to allow per-channel TIMING_UPDATE requests. Setting this DISABLED can cause ordering issue across the channels for the pad macro registers 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
10:9	WRITTEN	UPDATE_DLL_IN_UPDATE: [PMC] When to allow a pad macro autocal update to be triggered from a manual timing update. NEVER means it is never allowed. WRITTEN means allow the update if a pad macro autocal was written since the previous update. ALWAYS means to always issue the autocal update. 0 = NEVER 1 = WRITTEN 2 = ALWAYS
8	ENABLED	UPDATE_DLL_IN_CLKCHANGE: [PMC] Allow a pad macro DLL update to be triggered during a clock change timing update 0 = DISABLED 1 = ENABLED
2:1	NEVER	UPDATE_AUTO_CAL_IN_UPDATE: [PMC] When to allow a pad macro autocal update to be triggered from a manual timing update. NEVER means it is never allowed. WRITTEN means allow the update if a pad macro autocal was written since the previous update. ALWAYS means to always issue the autocal update. 0 = NEVER 1 = WRITTEN 2 = ALWAYS
0	ENABLED	UPDATE_AUTO_CAL_IN_CLKCHANGE: [PMC] Allow a pad macro autocal update to be triggered during a clock change timing update 0 = DISABLED 1 = ENABLED

18.11.2.222 EMC_PMACRO_QUSE_DDLL_RANK0_0_0

Rank0 QUSE DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x600 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK0_BYTE1: [PMC] This should be programmed to QUSE trimmer value for rank0 byte1
10:0	0x0	QUSE_DDLL_RANK0_BYTE0: [PMC] This should be programmed to QUSE trimmer value for rank0 byte0

18.11.2.223 EMC_PMACRO_QUSE_DDLL_RANK0_1_0

Rank0 QUSE DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x604 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK0_BYTE3: [PMC] This should be programmed to QUSE trimmer value for rank0 byte3
10:0	0x0	QUSE_DDLL_RANK0_BYTE2: [PMC] This should be programmed to QUSE trimmer value for rank0 byte2

18.11.2.224 EMC_PMACRO_QUSE_DDLL_RANK0_2_0

Rank0 QUSE DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x608 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK0_BYTE5: [PMC] This should be programmed to QUSE trimmer value for rank0 byte5
10:0	0x0	QUSE_DDLL_RANK0_BYTE4: [PMC] This should be programmed to QUSE trimmer value for rank0 byte4

18.11.2.225 EMC_PMACRO_QUSE_DDLL_RANK0_3_0

Rank0 QUSE DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x60c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK0_BYTE7: [PMC] This should be programmed to QUSE trimmer value for rank0 byte7
10:0	0x0	QUSE_DDLL_RANK0_BYTE6: [PMC] This should be programmed to QUSE trimmer value for rank0 byte6

18.11.2.226 EMC_PMACRO_QUSE_DDLL_RANK0_4_0

Rank0 CMD DDLL value for cmd1 and cmd0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x610 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK0_CMD1: [PMC] This should be programmed to QUSE trimmer value for rank0 cmd1
10:0	0x0	QUSE_DDLL_RANK0_CMD0: [PMC] This should be programmed to QUSE trimmer value for rank0 cmd0

18.11.2.227 EMC_PMACRO_QUSE_DDLL_RANK0_5_0

Rank0 CMD DDLL value for cmd2 and cmd3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x614 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK0_CMD3: [PMC] This should be programmed to QUSE trimmer value for rank0 cmd3
10:0	0x0	QUSE_DDLL_RANK0_CMD2: [PMC] This should be programmed to QUSE trimmer value for rank0 cmd2

18.11.2.228 EMC_PMACRO_QUSE_DDLL_RANK1_0_0

Rank1 QUSE DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x620 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK1_BYTE1: [PMC] This should be programmed to QUSE trimmer value for rank1 byte1
10:0	0x0	QUSE_DDLL_RANK1_BYTE0: [PMC] This should be programmed to QUSE trimmer value for rank1 byte0

18.11.2.229 EMC_PMACRO_QUSE_DDLL_RANK1_1_0

Rank1 QUSE DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x624 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK1_BYTE3: [PMC] This should be programmed to QUSE trimmer value for RANK1 byte3
10:0	0x0	QUSE_DDLL_RANK1_BYTE2: [PMC] This should be programmed to QUSE trimmer value for RANK1 byte2

18.11.2.230 EMC_PMACRO_QUSE_DDLL_RANK1_2_0

Rank1 QUSE DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x628 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK1_BYTE5: [PMC] This should be programmed to QUSE trimmer value for RANK1 byte5
10:0	0x0	QUSE_DDLL_RANK1_BYTE4: [PMC] This should be programmed to QUSE trimmer value for RANK1 byte4

18.11.2.231 EMC_PMACRO_QUSE_DDLL_RANK1_3_0

Rank1 QUSE DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x62c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK1_BYTE7: [PMC] This should be programmed to QUSE trimmer value for RANK1 byte7
10:0	0x0	QUSE_DDLL_RANK1_BYTE6: [PMC] This should be programmed to QUSE trimmer value for RANK1 byte6

18.11.2.232 EMC_PMACRO_QUSE_DDLL_RANK1_4_0

Rank1 CMD DDLL value for cmd0 and cmd1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x630 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK1_CMD1: [PMC] This should be programmed to QUSE trimmer value for RANK1 cmd1
10:0	0x0	QUSE_DDLL_RANK1_CMD0: [PMC] This should be programmed to QUSE trimmer value for RANK1 cmd0

18.11.2.233 EMC_PMACRO_QUSE_DDLL_RANK1_5_0

Rank1 CMD DDLL value for cmd2 and cmd3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x634 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	QUSE_DDLL_RANK1_CMD3: [PMC] This should be programmed to QUSE trimmer value for RANK1 cmd3

Bit	Reset	Description
10:0	0x0	QUSE_DDLL_RANK1_CMD2: [PMC] This should be programmed to QUSE trimmer value for RANK1 cmd2

18.11.2.234 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK0_0_0

Rank0 OB Long DQ DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x640 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE1: [PMC] This should be programmed to OB trimmer value for rank0 byte1
10:0	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE0: [PMC] This should be programmed to OB trimmer value for rank0 byte0

18.11.2.235 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK0_1_0

Rank0 OB Long DQ DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x644 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE3: [PMC] This should be programmed to OB trimmer value for RANK0 byte3
10:0	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE2: [PMC] This should be programmed to OB trimmer value for RANK0 byte2

18.11.2.236 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK0_2_0

Rank0 OB Long DQ DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x648 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE5: [PMC] This should be programmed to OB trimmer value for RANK0 byte5
10:0	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE4: [PMC] This should be programmed to OB trimmer value for RANK0 byte4

18.11.2.237 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK0_3_0

Rank0 OB Long DQ DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x64c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE7: [PMC] This should be programmed to OB trimmer value for RANK0 byte7
10:0	0x0	OB_DDLL_LONG_DQ_RANK0_BYTE6: [PMC] This should be programmed to OB trimmer value for RANK0 byte6

18.11.2.238 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK0_4_0

Rank0 OB Long DQ DDLL value for cmd0 and cmd1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x650 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK0_CMD1: [PMC] This should be programmed to OB trimmer value for RANK0 cmd1
10:0	0x0	OB_DDLL_LONG_DQ_RANK0_CMD0: [PMC] This should be programmed to OB trimmer value for RANK0 cmd0

18.11.2.239 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK0_5_0

Rank0 OB Long DQ DDLL value for cmd2 and cmd3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x654 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK0_CMD3: [PMC] This should be programmed to OB trimmer value for RANK0 cmd3
10:0	0x0	OB_DDLL_LONG_DQ_RANK0_CMD2: [PMC] This should be programmed to OB trimmer value for RANK0 cmd2

18.11.2.240 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK1_0_0

Rank1 OB Long DQ DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x660 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE1: [PMC] This should be programmed to OB trimmer value for rank1 byte1
10:0	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE0: [PMC] This should be programmed to OB trimmer value for rank1 byte0

18.11.2.241 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK1_1_0

Rank1 OB Long DQ DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x664 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE3: [PMC] This should be programmed to OB trimmer value for RANK1 byte3
10:0	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE2: [PMC] This should be programmed to OB trimmer value for RANK1 byte2

18.11.2.242 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK1_2_0

Rank1 OB Long DQ DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x668 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE5: [PMC] This should be programmed to OB trimmer value for RANK1 byte5
10:0	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE4: [PMC] This should be programmed to OB trimmer value for RANK1 byte4

18.11.2.243 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK1_3_0

Rank1 OB Long DQ DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x66c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE7: [PMC] This should be programmed to OB trimmer value for RANK1 byte7
10:0	0x0	OB_DDLL_LONG_DQ_RANK1_BYTE6: [PMC] This should be programmed to OB trimmer value for RANK1 byte6

18.11.2.244 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK1_4_0

Rank1 OB Long DQ DDLL value for cmd0 and cmd1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x670 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK1_CMD1: [PMC] This should be programmed to OB trimmer value for RANK1 cmd1
10:0	0x0	OB_DDLL_LONG_DQ_RANK1_CMD0: [PMC] This should be programmed to OB trimmer value for RANK1 cmd0

18.11.2.245 EMC_PMACRO_OB_DDLL_LONG_DQ_RANK1_5_0

Rank1 OB Long DQ DDLL value for cmd2 and cmd3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x674 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	OB_DDLL_LONG_DQ_RANK1_CMD3: [PMC] This should be programmed to OB trimmer value for RANK1 cmd3
10:0	0x0	OB_DDLL_LONG_DQ_RANK1_CMD2: [PMC] This should be programmed to OB trimmer value for RANK1 cmd2

18.11.2.246 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK0_0_0

Rank0 OB Long DQS DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x680 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE1: [PMC] This should be programmed to OB DQS trimmer value for rank0 byte1

Bit	Reset	Description
10:0	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE0: [PMC] This should be programmed to OB DQS trimmer value for rank0 byte0

18.11.2.247 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK0_1_0

Rank0 OB Long DQS DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x684 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE3: [PMC] This should be programmed to OB DQS trimmer value for RANK0 byte3
10:0	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE2: [PMC] This should be programmed to OB DQS trimmer value for RANK0 byte2

18.11.2.248 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK0_2_0

Rank0 OB Long DQS DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the
- Boot ROM during warm boot.

Offset: 0x688 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE5: [PMC] This should be programmed to OB DQS trimmer value for RANK0 byte5
10:0	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE4: [PMC] This should be programmed to OB DQS trimmer value for RANK0 byte4

18.11.2.249 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK0_3_0

Rank0 OB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x68c | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE7: [PMC] This should be programmed to OB DQS trimmer value for RANK0 byte7
10:0	0x20	OB_DDLL_LONG_DQS_RANK0_BYTE6: [PMC] This should be programmed to OB DQS trimmer value for RANK0 byte6

18.11.2.250 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK0_4_0

Rank0 OB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x690 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK0_CMD1: [PMC] This should be programmed to OB DQS trimmer value for RANK0 cmd1
10:0	0x20	OB_DDLL_LONG_DQS_RANK0_CMD0: [PMC] This should be programmed to OB DQS trimmer value for RANK0 cmd0

18.11.2.251 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK0_5_0

Rank0 OB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x694 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK0_CMD3: [PMC] This should be programmed to OB DQS trimmer value for RANK0 cmd1
10:0	0x20	OB_DDLL_LONG_DQS_RANK0_CMD2: [PMC] This should be programmed to OB DQS trimmer value for RANK0 cmd0

18.11.2.252 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK1_0_0

Rank1 OB Long DQS DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6a0 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE1: [PMC] This should be programmed to OB DQS trimmer value for rank0 byte1
10:0	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE0: [PMC] This should be programmed to OB DQS trimmer value for rank0 byte0

18.11.2.253 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK1_1_0

Rank1 OB Long DQS DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6a4 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE3: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte3
10:0	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE2: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte2

18.11.2.254 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK1_2_0

Rank1 OB Long DQS DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6a8 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE5: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte5
10:0	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE4: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte4

18.11.2.255 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK1_3_0

Rank1 OB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6ac | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE7: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte7
10:0	0x20	OB_DDLL_LONG_DQS_RANK1_BYTE6: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte6

18.11.2.256 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK1_4_0

Rank1 OB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6b0 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK1_CMD1: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte7
10:0	0x20	OB_DDLL_LONG_DQS_RANK1_CMD0: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte6

18.11.2.257 EMC_PMACRO_OB_DDLL_LONG_DQS_RANK1_5_0

Rank1 OB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6b4 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	OB_DDLL_LONG_DQS_RANK1_CMD3: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte7
10:0	0x20	OB_DDLL_LONG_DQS_RANK1_CMD2: [PMC] This should be programmed to OB DQS trimmer value for RANK1 byte6

18.11.2.258 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK0_0_0

Rank0 IB Long DQS DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6c0 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE1: [PMC] This should be programmed to IB DQS trimmer value for rank0 byte1
10:0	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE0: [PMC] This should be programmed to IB DQS trimmer value for rank0 byte0

18.11.2.259 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK0_1_0

Rank0 IB Long DQS DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6c4 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE3: [PMC] This should be programmed to IB DQS trimmer value for RANK0 byte3

Bit	Reset	Description
10:0	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE2: [PMC] This should be programmed to IB DQS trimmer value for RANK0 byte2

18.11.2.260 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK0_2_0

Rank0 IB Long DQS DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6c8 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE5: [PMC] This should be programmed to IB DQS trimmer value for RANK0 byte5
10:0	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE4: [PMC] This should be programmed to IB DQS trimmer value for RANK0 byte4

18.11.2.261 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK0_3_0

Rank0 IB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6cc | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE7: [PMC] This should be programmed to IB DQS trimmer value for RANK0 byte7
10:0	0x20	IB_DDLL_LONG_DQS_RANK0_BYTE6: [PMC] This should be programmed to IB DQS trimmer value for RANK0 byte6

18.11.2.262 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK0_4_0

Rank0 IB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x6d0 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK0_CMD1: This should be programmed to IB DQS trimmer value for RANK0 byte7
10:0	0x20	IB_DDLL_LONG_DQS_RANK0_CMD0: This should be programmed to IB DQS trimmer value for RANK0 byte6

18.11.2.263 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK0_5_0

Rank0 IB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x6d4 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK0_CMD3: This should be programmed to IB DQS trimmer value for RANK0 byte7
10:0	0x20	IB_DDLL_LONG_DQS_RANK0_CMD2: This should be programmed to IB DQS trimmer value for RANK0 byte6

18.11.2.264 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK1_0_0

Rank1 IB Long DQS DDLL value for Byte0 and Byte1

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6e0 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE1: [PMC] This should be programmed to IB DQS trimmer value for rank1 byte1
10:0	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE0: [PMC] This should be programmed to IB DQS trimmer value for rank1 byte0

18.11.2.265 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK1_1_0

Rank1 IB Long DQS DDLL value for Byte2 and Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6e4 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE3: [PMC] This should be programmed to IB DQS trimmer value for RANK1 byte3
10:0	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE2: [PMC] This should be programmed to IB DQS trimmer value for RANK1 byte2

18.11.2.266 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK1_2_0

Rank1 IB Long DQS DDLL value for Byte4 and Byte5

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the
- Boot ROM during warm boot.

Offset: 0x6e8 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE5: [PMC] This should be programmed to IB DQS trimmer value for RANK1 byte5
10:0	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE4: [PMC] This should be programmed to IB DQS trimmer value for RANK1 byte4

18.11.2.267 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK1_3_0

Rank1 IB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6ec | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE7: [PMC] This should be programmed to IB DQS trimmer value for RANK1 byte7
10:0	0x20	IB_DDLL_LONG_DQS_RANK1_BYTE6: [PMC] This should be programmed to IB DQS trimmer value for RANK1 byte6

18.11.2.268 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK1_4_0

Rank1 IB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x6f0 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK1_CMD1: This should be programmed to IB DQS trimmer value for RANK1 byte7
10:0	0x20	IB_DDLL_LONG_DQS_RANK1_CMD0: This should be programmed to IB DQS trimmer value for RANK1 byte6

18.11.2.269 EMC_PMACRO_IB_DDLL_LONG_DQS_RANK1_5_0

Rank1 IB Long DQS DDLL value for Byte6 and Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x6f4 | Read/Write: R/W | Reset: 0x00200020 (0bxxxxx00000100000xxxxx00000100000)

Bit	Reset	Description
26:16	0x20	IB_DDLL_LONG_DQS_RANK1_CMD3: This should be programmed to IB DQS trimmer value for RANK1 byte7
10:0	0x20	IB_DDLL_LONG_DQS_RANK1_CMD2: This should be programmed to IB DQS trimmer value for RANK1 byte6

18.11.2.270 EMC_PMACRO_AUTOCAL_CFG_0_0

Autocal padmacro register for controls that are per-IOBRICK

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x700 | Read/Write: R/W | Reset: 0x04040404 (0bxxxxx100xxxxx100xxxxx100xxxxx100)

Bit	Reset	Description
26	ENABLE	DQS_E_CAL_BYPASS_BYTE3: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
25	DQS	DQS_DRV_SEL_BYTE3: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
24	CMD	CMD_DRV_SEL_BYTE3: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
18	ENABLE	DQS_E_CAL_BYPASS_BYTE2: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
17	DQS	DQS_DRV_SEL_BYTE2: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
16	CMD	CMD_DRV_SEL_BYTE2: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
10	ENABLE	DQS_E_CAL_BYPASS_BYTE1: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
9	DQS	DQS_DRV_SEL_BYTE1: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
8	CMD	CMD_DRV_SEL_BYTE1: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
2	ENABLE	DQS_E_CAL_BYPASS_BYTE0: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
1	DQS	DQS_DRV_SEL_BYTE0: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
0	CMD	CMD_DRV_SEL_BYTE0: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ cal codes 0 = CMD 1 = CA

18.11.2.271 EMC_PMACRO_AUTOCAL_CFG_1_0

Autocal padmacro register for controls that are per-IOBRICK

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x704 | Read/Write: R/W | Reset: 0x04040404 (0bxxxxx100xxxxx100xxxxx100xxxxx100)

Bit	Reset	Description
26	ENABLE	DQS_E_CAL_BYPASS_BYTE7: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
25	DQS	DQS_DRV_SEL_BYTE7: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
24	CMD	CMD_DRV_SEL_BYTE7: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
18	ENABLE	DQS_E_CAL_BYPASS_BYTE6: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
17	DQS	DQS_DRV_SEL_BYTE6: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
16	CMD	CMD_DRV_SEL_BYTE6: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
10	ENABLE	DQS_E_CAL_BYPASS_BYTE5: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
9	DQS	DQS_DRV_SEL_BYTE5: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
8	CMD	CMD_DRV_SEL_BYTE5: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
2	ENABLE	DQS_E_CAL_BYPASS_BYTE4: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
1	DQS	DQS_DRV_SEL_BYTE4: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
0	CMD	CMD_DRV_SEL_BYTE4: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA

18.11.2.272 EMC_PMACRO_AUTOCAL_CFG_2_0

Autocal pad macro register for controls that are per-IOBRICK

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x708 | Read/Write: R/W | Reset: 0x04040404 (0bxxxxx100xxxxx100xxxxx100xxxxx100)

Bit	Reset	Description
26	ENABLE	DQS_E_CAL_BYPASS_CMD3: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
25	DQS	DQS_DRV_SEL_CMD3: [PMC] DQS=DQS pad uses DQS/CK cal codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
24	CMD	CMD_DRV_SEL_CMD3: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ cal codes 0 = CMD 1 = CA
18	ENABLE	DQS_E_CAL_BYPASS_CMD2: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
17	DQS	DQS_DRV_SEL_CMD2: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
16	CMD	CMD_DRV_SEL_CMD2: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
10	ENABLE	DQS_E_CAL_BYPASS_CMD1: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
9	DQS	DQS_DRV_SEL_CMD1: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
8	CMD	CMD_DRV_SEL_CMD1: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA
2	ENABLE	DQS_E_CAL_BYPASS_CMD0: [PMC] If enabled, ignore E_CAL_UPDATE pulse and apply DRVUP/DRVDN calibration codes immediately 0 = DISABLE 1 = ENABLE
1	DQS	DQS_DRV_SEL_CMD0: [PMC] DQS=DQS pad uses DQS/CK calibration codes based on IOBRICK config. CA=DQS pad uses CA/DQ calibration codes 0 = DQS 1 = CA
0	CMD	CMD_DRV_SEL_CMD0: [PMC] CMD=CMD pad uses CMD calibration codes. CA=CMD pad uses CA/DQ calibration codes 0 = CMD 1 = CA

18.11.2.273 EMC_PMACRO_TX_PWRD_0_0

Clock gate register for unused TX bits in byte0 and byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x720 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29	0x0	CMD_TX_E_WKPD_BYTE1: [PMC] Active high to enable weak pull down for CMD pin of byte1 brick
28	0x0	CMD_TX_E_WKPU_BYTE1: [PMC] Active high to enable weak pull up for CMD pin of byte1 brick
27	0x0	CMD_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for CMD pin of byte1 brick
26	0x0	DQSN_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQSN of byte1
25	0x0	DQSP_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQSP of byte1
24	0x0	DQ8_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte1
23	0x0	DQ7_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte1
22	0x0	DQ6_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte1
21	0x0	DQ5_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte1
20	0x0	DQ4_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte1
19	0x0	DQ3_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte1
18	0x0	DQ2_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte1
17	0x0	DQ1_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte1
16	0x0	DQ0_TX_PWRD_BYTE1: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte1
13	0x0	CMD_TX_E_WKPD_BYTE0: [PMC] Active high to enable weak pull down for CMD pin of byte0 brick
12	0x0	CMD_TX_E_WKPU_BYTE0: [PMC] Active high to enable weak pull up for CMD pin of byte0 brick
11	0x0	CMD_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for CMD pin of byte0 brick
10	0x0	DQSN_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQSN of byte0
9	0x0	DQSP_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQSP of byte0
8	0x0	DQ8_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte0
7	0x0	DQ7_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte0
6	0x0	DQ6_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte0
5	0x0	DQ5_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte0
4	0x0	DQ4_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte0
3	0x0	DQ3_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte0
2	0x0	DQ2_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte0
1	0x0	DQ1_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte0
0	0x0	DQ0_TX_PWRD_BYTE0: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte0

18.11.2.274 EMC_PMACRO_TX_PWRD_1_0

Clock gate register for unused TX bits in byte2 and byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x724 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29	0x0	CMD_TX_E_WKPD_BYTE3: [PMC] Active high to enable weak pull down for CMD pin of byte3 brick
28	0x0	CMD_TX_E_WKPU_BYTE3: [PMC] Active high to enable weak pull up for CMD pin of byte3 brick
27	0x0	CMD_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for CMD pin of byte3 brick
26	0x0	DQSN_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQSN of byte3
25	0x0	DQSP_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQSP of byte3

Bit	Reset	Description
24	0x0	DQ8_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte3
23	0x0	DQ7_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte3
22	0x0	DQ6_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte3
21	0x0	DQ5_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte3
20	0x0	DQ4_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte3
19	0x0	DQ3_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte3
18	0x0	DQ2_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte3
17	0x0	DQ1_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte3
16	0x0	DQ0_TX_PWRD_BYTE3: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte3
13	0x0	CMD_TX_E_WKPD_BYTE2: [PMC] Active high to enable weak pull down for CMD pin of byte2 brick
12	0x0	CMD_TX_E_WKPU_BYTE2: [PMC] Active high to enable weak pull up for CMD pin of byte2 brick
11	0x0	CMD_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for CMD pin of byte2 brick
10	0x0	DQSN_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQSN of byte2
9	0x0	DQSP_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQSP of byte2
8	0x0	DQ8_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte2
7	0x0	DQ7_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte2
6	0x0	DQ6_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte2
5	0x0	DQ5_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte2
4	0x0	DQ4_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte2
3	0x0	DQ3_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte2
2	0x0	DQ2_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte2
1	0x0	DQ1_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte2
0	0x0	DQ0_TX_PWRD_BYTE2: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte2

18.11.2.275 EMC_PMACRO_TX_PWRD_2_0

Clock gate register for unused TX bits in byte5 and byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x728 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29	0x0	CMD_TX_E_WKPD_BYTE5: [PMC] Active high to enable weak pull down for CMD pin of byte5 brick
28	0x0	CMD_TX_E_WKPU_BYTE5: [PMC] Active high to enable weak pull up for CMD pin of byte5 brick
27	0x0	CMD_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for CMD pin of byte5 brick
26	0x0	DQSN_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQSN of byte5
25	0x0	DQSP_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQSP of byte5
24	0x0	DQ8_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte5
23	0x0	DQ7_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte5
22	0x0	DQ6_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte5
21	0x0	DQ5_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte5
20	0x0	DQ4_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte5
19	0x0	DQ3_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte5
18	0x0	DQ2_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte5

Bit	Reset	Description
17	0x0	DQ1_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte5
16	0x0	DQ0_TX_PWRD_BYTE5: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte5
13	0x0	CMD_TX_E_WKPD_BYTE4: [PMC] Active high to enable weak pull down for CMD pin of byte4 brick
12	0x0	CMD_TX_E_WKPU_BYTE4: [PMC] Active high to enable weak pull up for CMD pin of byte4 brick
11	0x0	CMD_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for CMD pin of byte4 brick
10	0x0	DQSN_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQSN of byte4
9	0x0	DQSP_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQSP of byte4
8	0x0	DQ8_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte4
7	0x0	DQ7_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte4
6	0x0	DQ6_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte4
5	0x0	DQ5_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte4
4	0x0	DQ4_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte4
3	0x0	DQ3_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte4
2	0x0	DQ2_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte4
1	0x0	DQ1_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte4
0	0x0	DQ0_TX_PWRD_BYTE4: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte4

18.11.2.276 EMC_PMACRO_TX_PWRD_3_0

Clock gate register for unused TX bits in byte6 and byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x72c | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29	0x0	CMD_TX_E_WKPD_BYTE7: [PMC] Active high to enable weak pull down for CMD pin of byte7 brick
28	0x0	CMD_TX_E_WKPU_BYTE7: [PMC] Active high to enable weak pull up for CMD pin of byte7 brick
27	0x0	CMD_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for CMD pin of byte7 brick
26	0x0	DQSN_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQSN of byte7
25	0x0	DQSP_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQSP of byte7
24	0x0	DQ8_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte7
23	0x0	DQ7_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte7
22	0x0	DQ6_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte7
21	0x0	DQ5_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte7
20	0x0	DQ4_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte7
19	0x0	DQ3_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte7
18	0x0	DQ2_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte7
17	0x0	DQ1_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte7
16	0x0	DQ0_TX_PWRD_BYTE7: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte7
13	0x0	CMD_TX_E_WKPD_BYTE6: [PMC] Active high to enable weak pull down for CMD pin of byte6 brick
12	0x0	CMD_TX_E_WKPU_BYTE6: [PMC] Active high to enable weak pull up for CMD pin of byte6 brick
11	0x0	CMD_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for CMD pin of byte6 brick
10	0x0	DQSN_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQSN of byte6

Bit	Reset	Description
9	0x0	DQSP_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQSP of byte6
8	0x0	DQ8_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit8 of byte6
7	0x0	DQ7_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit7 of byte6
6	0x0	DQ6_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit6 of byte6
5	0x0	DQ5_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit5 of byte6
4	0x0	DQ4_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit4 of byte6
3	0x0	DQ3_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit3 of byte6
2	0x0	DQ2_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit2 of byte6
1	0x0	DQ1_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit1 of byte6
0	0x0	DQ0_TX_PWRD_BYTE6: [PMC] This should be programmed to TX PWRD value for DQ bit0 of byte6

18.11.2.277 EMC_PMACRO_TX_PWRD_4_0

Clock gate register for unused TX bits in cmd0 and cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x730 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29	0x0	CMD_TX_E_WKPD_CMD1: [PMC] Active high to enable weak pull down for CMD pin of CMD1 brick
28	0x0	CMD_TX_E_WKPU_CMD1: [PMC] Active high to enable weak pull up for CMD pin of CMD1 brick
27	0x0	CMD_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for CMD pin of CMD1 brick
26	0x0	DQSN_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQSN of CMD1
25	0x0	DQSP_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQSP of CMD1
24	0x0	DQ8_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit8 of CMD1
23	0x0	DQ7_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit7 of CMD1
22	0x0	DQ6_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit6 of CMD1
21	0x0	DQ5_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit5 of CMD1
20	0x0	DQ4_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit4 of CMD1
19	0x0	DQ3_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit3 of CMD1
18	0x0	DQ2_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit2 of CMD1
17	0x0	DQ1_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit1 of CMD1
16	0x0	DQ0_TX_PWRD_CMD1: [PMC] This should be programmed to TX PWRD value for DQ bit0 of CMD1
13	0x0	CMD_TX_E_WKPD_CMD0: [PMC] Active high to enable weak pull down for CMD pin of CMD0 brick
12	0x0	CMD_TX_E_WKPU_CMD0: [PMC] Active high to enable weak pull up for CMD pin of CMD0 brick
11	0x0	CMD_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for CMD pin of CMD0 brick
10	0x0	DQSN_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQSN of CMD0
9	0x0	DQSP_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQSP of CMD0
8	0x0	DQ8_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit8 of CMD0
7	0x0	DQ7_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit7 of CMD0
6	0x0	DQ6_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit6 of CMD0
5	0x0	DQ5_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit5 of CMD0
4	0x0	DQ4_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit4 of CMD0
3	0x0	DQ3_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit3 of CMD0

2	0x0	DQ2_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit2 of CMD0
1	0x0	DQ1_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit1 of CMD0
0	0x0	DQ0_TX_PWRD_CMD0: [PMC] This should be programmed to TX PWRD value for DQ bit0 of CMD0

18.11.2.278 EMC_PMACRO_TX_PWRD_5_0

Clock gate register for unused TX bits in cmd2 and cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x734 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29	0x0	CMD_TX_E_WKPD_CMD3: [PMC] Active high to enable weak pull down for CMD pin of CMD3 brick
28	0x0	CMD_TX_E_WKPU_CMD3: [PMC] Active high to enable weak pull up for CMD pin of CMD3 brick
27	0x0	CMD_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for CMD pin of CMD3 brick
26	0x0	DQSN_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQSN of CMD3
25	0x0	DQSP_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQSP of CMD3
24	0x0	DQ8_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit8 of CMD3
23	0x0	DQ7_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit7 of CMD3
22	0x0	DQ6_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit6 of CMD3
21	0x0	DQ5_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit5 of CMD3
20	0x0	DQ4_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit4 of CMD3
19	0x0	DQ3_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit3 of CMD3
18	0x0	DQ2_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit2 of CMD3
17	0x0	DQ1_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit1 of CMD3
16	0x0	DQ0_TX_PWRD_CMD3: [PMC] This should be programmed to TX PWRD value for DQ bit0 of CMD3
13	0x0	CMD_TX_E_WKPD_CMD2: [PMC] Active high to enable weak pull down for CMD pin of CMD2 brick
12	0x0	CMD_TX_E_WKPU_CMD2: [PMC] Active high to enable weak pull up for CMD pin of CMD2 brick
11	0x0	CMD_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for CMD pin of CMD2 brick
10	0x0	DQSN_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQSN of CMD2
9	0x0	DQSP_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQSP of CMD2
8	0x0	DQ8_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit8 of CMD2
7	0x0	DQ7_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit7 of CMD2
6	0x0	DQ6_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit6 of CMD2
5	0x0	DQ5_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit5 of CMD2
4	0x0	DQ4_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit4 of CMD2
3	0x0	DQ3_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit3 of CMD2
2	0x0	DQ2_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit2 of CMD2
1	0x0	DQ1_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit1 of CMD2
0	0x0	DQ0_TX_PWRD_CMD2: [PMC] This should be programmed to TX PWRD value for DQ bit0 of CMD2

18.11.2.279 EMC_PMACRO_TX_SEL_CLK_SRC_0_0

Select clock source for 0.5T and 1.0T when brick is configured as ADDR/CMD brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x740 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	CMD_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte1 brick
26	0x0	DQSN_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQSN of byte1
25	0x0	DQSP_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQSP of byte1
24	0x0	DQ8_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte1
23	0x0	DQ7_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte1
22	0x0	DQ6_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte1
21	0x0	DQ5_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte1
20	0x0	DQ4_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte1
19	0x0	DQ3_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte1
18	0x0	DQ2_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte1
17	0x0	DQ1_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte1
16	0x0	DQ0_TX_SEL_CLK_SRC_BYTE1: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte1
11	0x0	CMD_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte0 brick
10	0x0	DQSN_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQSN of byte0
9	0x0	DQSP_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQSP of byte0
8	0x0	DQ8_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte0
7	0x0	DQ7_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte0
6	0x0	DQ6_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte0
5	0x0	DQ5_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte0
4	0x0	DQ4_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte0
3	0x0	DQ3_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte0
2	0x0	DQ2_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte0
1	0x0	DQ1_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte0
0	0x0	DQ0_TX_SEL_CLK_SRC_BYTE0: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte0

18.11.2.280 EMC_PMACRO_TX_SEL_CLK_SRC_1_0

Select clock source for 0.5T and 1.0T when brick is configured as ADDR/CMD brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x744 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	CMD_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte3 brick
26	0x0	DQSN_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQSN of byte3
25	0x0	DQSP_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQSP of byte3
24	0x0	DQ8_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte3
23	0x0	DQ7_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte3
22	0x0	DQ6_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte3
21	0x0	DQ5_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte3
20	0x0	DQ4_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte3
19	0x0	DQ3_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte3
18	0x0	DQ2_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte3
17	0x0	DQ1_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte3
16	0x0	DQ0_TX_SEL_CLK_SRC_BYTE3: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte3
11	0x0	CMD_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte2 brick
10	0x0	DQSN_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQSN of byte2
9	0x0	DQSP_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQSP of byte2
8	0x0	DQ8_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte2
7	0x0	DQ7_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte2
6	0x0	DQ6_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte2
5	0x0	DQ5_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte2
4	0x0	DQ4_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte2
3	0x0	DQ3_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte2
2	0x0	DQ2_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte2
1	0x0	DQ1_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte2

Bit	Reset	Description
0	0x0	DQ0_TX_SEL_CLK_SRC_BYTE2: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte2

18.11.2.281 EMC_PMACRO_TX_SEL_CLK_SRC_2_0

Select clock source for 0.5T and 1.0T when brick is configured as ADDR/CMD brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x748 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	CMD_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte5 brick
26	0x0	DQSN_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQSN of byte5
25	0x0	DQSP_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQSP of byte5
24	0x0	DQ8_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte5
23	0x0	DQ7_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte5
22	0x0	DQ6_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte5
21	0x0	DQ5_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte5
20	0x0	DQ4_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte5
19	0x0	DQ3_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte5
18	0x0	DQ2_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte5
17	0x0	DQ1_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte5
16	0x0	DQ0_TX_SEL_CLK_SRC_BYTE5: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte5
11	0x0	CMD_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte4 brick
10	0x0	DQSN_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQSN of byte4
9	0x0	DQSP_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQSP of byte4
8	0x0	DQ8_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte4
7	0x0	DQ7_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte4
6	0x0	DQ6_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte4
5	0x0	DQ5_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte4
4	0x0	DQ4_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte4
3	0x0	DQ3_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte4

Bit	Reset	Description
2	0x0	DQ2_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte4
1	0x0	DQ1_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte4
0	0x0	DQ0_TX_SEL_CLK_SRC_BYTE4: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte4

18.11.2.282 EMC_PMACRO_TX_SEL_CLK_SRC_3_0

Select clock source for 0.5T and 1.0T when brick is configured as ADDR/CMD brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x74c | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	CMD_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte7 brick
26	0x0	DQSN_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQSN of byte7
25	0x0	DQSP_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQSP of byte7
24	0x0	DQ8_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte7
23	0x0	DQ7_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte7
22	0x0	DQ6_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte7
21	0x0	DQ5_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte7
20	0x0	DQ4_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte7
19	0x0	DQ3_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte7
18	0x0	DQ2_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte7
17	0x0	DQ1_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte7
16	0x0	DQ0_TX_SEL_CLK_SRC_BYTE7: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte7
11	0x0	CMD_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for CMD pin of byte6 brick
10	0x0	DQSN_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQSN of byte6
9	0x0	DQSP_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQSP of byte6
8	0x0	DQ8_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of byte6
7	0x0	DQ7_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of byte6
6	0x0	DQ6_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of byte6
5	0x0	DQ5_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of byte6

Bit	Reset	Description
4	0x0	DQ4_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of byte6
3	0x0	DQ3_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of byte6
2	0x0	DQ2_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of byte6
1	0x0	DQ1_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of byte6
0	0x0	DQ0_TX_SEL_CLK_SRC_BYTE6: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of byte6

18.11.2.283 EMC_PMACRO_TX_SEL_CLK_SRC_4_0

Select clock source for 0.5T and 1.0T when brick is configured as ADDR/CMD brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x750 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	CMD_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for CMD pin of CMD1 brick
26	0x0	DQSN_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQSN of CMD1
25	0x0	DQSP_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQSP of CMD1
24	0x0	DQ8_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of CMD1
23	0x0	DQ7_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of CMD1
22	0x0	DQ6_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of CMD1
21	0x0	DQ5_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of CMD1
20	0x0	DQ4_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of CMD1
19	0x0	DQ3_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of CMD1
18	0x0	DQ2_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of CMD1
17	0x0	DQ1_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of CMD1
16	0x0	DQ0_TX_SEL_CLK_SRC_CMD1: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of CMD1
11	0x0	CMD_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for CMD pin of CMD0 brick
10	0x0	DQSN_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQSN of CMD0
9	0x0	DQSP_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQSP of CMD0
8	0x0	DQ8_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of CMD0
7	0x0	DQ7_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of CMD0

Bit	Reset	Description
6	0x0	DQ6_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of CMD0
5	0x0	DQ5_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of CMD0
4	0x0	DQ4_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of CMD0
3	0x0	DQ3_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of CMD0
2	0x0	DQ2_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of CMD0
1	0x0	DQ1_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of CMD0
0	0x0	DQ0_TX_SEL_CLK_SRC_CMD0: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of CMD0

18.11.2.284 EMC_PMACRO_TX_SEL_CLK_SRC_5_0

Select clock source for 0.5T and 1.0T when brick is configured as ADDR/CMD brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x754 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxx000000000000)

Bit	Reset	Description
27	0x0	CMD_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for CMD pin of CMD3 brick
26	0x0	DQSN_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQSN of CMD3
25	0x0	DQSP_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQSP of CMD3
24	0x0	DQ8_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of CMD3
23	0x0	DQ7_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of CMD3
22	0x0	DQ6_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of CMD3
21	0x0	DQ5_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of CMD3
20	0x0	DQ4_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of CMD3
19	0x0	DQ3_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of CMD3
18	0x0	DQ2_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of CMD3
17	0x0	DQ1_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of CMD3
16	0x0	DQ0_TX_SEL_CLK_SRC_CMD3: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of CMD3
11	0x0	CMD_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for CMD pin of CMD2 brick
10	0x0	DQSN_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQSN of CMD2
9	0x0	DQSP_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQSP of CMD2

Bit	Reset	Description
8	0x0	DQ8_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit8 of CMD2
7	0x0	DQ7_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit7 of CMD2
6	0x0	DQ6_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit6 of CMD2
5	0x0	DQ5_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit5 of CMD2
4	0x0	DQ4_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit4 of CMD2
3	0x0	DQ3_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit3 of CMD2
2	0x0	DQ2_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit2 of CMD2
1	0x0	DQ1_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit1 of CMD2
0	0x0	DQ0_TX_SEL_CLK_SRC_CMD2: [PMC] This should be programmed to TX clock source sel value for DQ bit0 of CMD2

18.11.2.285 EMC_PMACRO_DDLL_BYPASS_0

DLL and DDLLCAL bypass

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x760 | Read/Write: R/W | Reset: 0x00ff00ff (0b000x000011111111000x000011111111)

Bit	Reset	Description
31	0x0	CMD_DDLLCAL_BIT_RXDQ_BYPASS: [PMC] Bypass the DDLL calibration value
30	0x0	CMD_DDLLCAL_BIT_TXDQ_BYPASS: [PMC] Bypass the DDLL calibration value
29	0x0	CMD_DDLLCAL_BIT_TXDQS_BYPASS: [PMC] Bypass the DDLL calibration value
27	0x0	CMD_DDLLCAL_BYTE_QU_BYPASS: [PMC] Bypass the DDLL calibration value
26	0x0	CMD_DDLLCAL_BYTE_RXDQS_BYPASS: [PMC] Bypass the DDLL calibration value
25	0x0	CMD_DDLLCAL_BYTE_TXDQS_BYPASS: [PMC] Bypass the DDLL calibration value
24	0x0	CMD_DDLLCAL_BYTE_TXDQ_BYPASS: [PMC] Bypass the DDLL calibration value
23	0x1	CMD_DDLL_BIT_RXDQ_BYPASS: [PMC] This should be programmed to RX fine trimmer bypass value for DQS
22	0x1	CMD_DDLL_BIT_TXDQS_BYPASS: [PMC] This should be programmed to TX fine trimmer bypass value for DQS
21	0x1	CMD_DDLL_BIT_TXDQ_BYPASS: [PMC] This should be programmed to TX fine trimmer bypass value for DQ
20	0x1	CMD_DDLL_BYTE_RXRT_BYPASS: [PMC] This should be programmed to RX trimmer bypass value for retiming flop
19	0x1	CMD_DDLL_BYTE_QU_BYPASS: [PMC] This should be programmed to QUSE trimmer bypass value for
18	0x1	CMD_DDLL_BYTE_RXDQS_BYPASS: [PMC] This should be programmed to RX trimmer bypass value for DQS
17	0x1	CMD_DDLL_BYTE_TXDQS_BYPASS: [PMC] This should be programmed to TX trimmer bypass value for DQS
16	0x1	CMD_DDLL_BYTE_TXDQ_BYPASS: [PMC] This should be programmed to TX trimmer bypass value for DQ. 0= TRIM mode, 1= Bypass fine interpolator
15	0x0	DATA_DDLLCAL_BIT_RXDQ_BYPASS: [PMC] Bypass the DDLL calibration value
14	0x0	DATA_DDLLCAL_BIT_TXDQ_BYPASS: [PMC] Bypass the DDLL calibration value

Bit	Reset	Description
13	0x0	DATA_DDLLCAL_BIT_TXDQS_BYPASS: [PMC] Bypass the DDLL calibration value
11	0x0	DATA_DDLLCAL_BYTE_QU_BYPASS: [PMC] Bypass the DDLL calibration value
10	0x0	DATA_DDLLCAL_BYTE_RXDQS_BYPASS: [PMC] Bypass the DDLL calibration value
9	0x0	DATA_DDLLCAL_BYTE_TXDQS_BYPASS: [PMC] Bypass the DDLL calibration value
8	0x0	DATA_DDLLCAL_BYTE_TXDQ_BYPASS: [PMC] Bypass the DDLL calibration value
7	0x1	DATA_DDLL_BIT_RXDQ_BYPASS: [PMC] This should be programmed to RX fine trimmer bypass value for DQS
6	0x1	DATA_DDLL_BIT_TXDQS_BYPASS: [PMC] This should be programmed to TX fine trimmer bypass value for DQS
5	0x1	DATA_DDLL_BIT_TXDQ_BYPASS: [PMC] This should be programmed to TX fine trimmer bypass value for DQ
4	0x1	DATA_DDLL_BYTE_RXRT_BYPASS: [PMC] This should be programmed to RX trimmer bypass value for retiming flop
3	0x1	DATA_DDLL_BYTE_QU_BYPASS: [PMC] This should be programmed to QUSE trimmer bypass value for
2	0x1	DATA_DDLL_BYTE_RXDQS_BYPASS: [PMC] This should be programmed to RX trimmer bypass value for DQS
1	0x1	DATA_DDLL_BYTE_TXDQS_BYPASS: [PMC] This should be programmed to TX trimmer bypass value for DQS
0	0x1	DATA_DDLL_BYTE_TXDQ_BYPASS: [PMC] This should be programmed to TX trimmer bypass value for DQ. 0= TRIM mode, 1= Bypass fine interpolator

18.11.2.286 EMC_PMACRO_DDLL_PWRD_0_0

DLL power down mode enables

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x770 | Read/Write: R/W | Reset: 0x94949494 (0b10x1010010x1010010x1010010x10100)

Bit	Reset	Description
31	0x1	DDLL_BIT_RXDQ_PWRD_BYTE3: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
30	0x0	DDLL_BIT_TXDQ_PWRD_BYTE3: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
28	0x1	DDLL_BYTE_RXRT_PWRD_BYTE3: [PMC] This should be programmed to RX power down mode enable for retiming flop
27	0x0	DDLL_BYTE_QU_PWRD_BYTE3: [PMC] This should be programmed to QUSE power down mode enable for
26	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE3: [PMC] This should be programmed to RX power down mode enable for DQS
25	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE3: [PMC] This should be programmed to TX power down mode enable for DQS
24	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE3: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
23	0x1	DDLL_BIT_RXDQ_PWRD_BYTE2: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
22	0x0	DDLL_BIT_TXDQ_PWRD_BYTE2: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
20	0x1	DDLL_BYTE_RXRT_PWRD_BYTE2: [PMC] This should be programmed to RX power down mode enable for retiming flop
19	0x0	DDLL_BYTE_QU_PWRD_BYTE2: [PMC] This should be programmed to QUSE power down mode enable for

18	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE2: [PMC] This should be programmed to RX power down mode enable for DQS
17	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE2: [PMC] This should be programmed to TX power down mode enable for DQS
16	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE2: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
15	0x1	DDLL_BIT_RXDQ_PWRD_BYTE1: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
14	0x0	DDLL_BIT_TXDQ_PWRD_BYTE1: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
12	0x1	DDLL_BYTE_RXRT_PWRD_BYTE1: [PMC] This should be programmed to RX power down mode enable for retiming flop
11	0x0	DDLL_BYTE_QU_PWRD_BYTE1: [PMC] This should be programmed to QUSE power down mode enable for
10	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE1: [PMC] This should be programmed to RX power down mode enable for DQS
9	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE1: [PMC] This should be programmed to TX power down mode enable for DQS
8	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE1: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
7	0x1	DDLL_BIT_RXDQ_PWRD_BYTE0: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
6	0x0	DDLL_BIT_TXDQ_PWRD_BYTE0: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
4	0x1	DDLL_BYTE_RXRT_PWRD_BYTE0: [PMC] This should be programmed to RX power down mode enable for retiming flop
3	0x0	DDLL_BYTE_QU_PWRD_BYTE0: [PMC] This should be programmed to QUSE power down mode enable for
2	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE0: [PMC] This should be programmed to RX power down mode enable for DQS
1	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE0: [PMC] This should be programmed to TX power down mode enable for DQS
0	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE0: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping

18.11.2.287 EMC_PMACRO_DDLL_PWRD_1_0

DLL power down mode enables

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x774 | Read/Write: R/W | Reset: 0x94949494 (0b10x1010010x1010010x1010010x10100)

Bit	Reset	Description
31	0x1	DDLL_BIT_RXDQ_PWRD_BYTE7: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
30	0x0	DDLL_BIT_TXDQ_PWRD_BYTE7: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
28	0x1	DDLL_BYTE_RXRT_PWRD_BYTE7: [PMC] This should be programmed to RX power down mode enable for retiming flop
27	0x0	DDLL_BYTE_QU_PWRD_BYTE7: [PMC] This should be programmed to QUSE power down mode enable for
26	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE7: [PMC] This should be programmed to RX power down mode enable for DQS

Bit	Reset	Description
25	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE7: [PMC] This should be programmed to TX power down mode enable for DQS
24	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE7: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
23	0x1	DDLL_BIT_RXDQ_PWRD_BYTE6: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
22	0x0	DDLL_BIT_TXDQ_PWRD_BYTE6: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
20	0x1	DDLL_BYTE_RXRT_PWRD_BYTE6: [PMC] This should be programmed to RX power down mode enable for retiming flop
19	0x0	DDLL_BYTE_QU_PWRD_BYTE6: [PMC] This should be programmed to QUSE power down mode enable for
18	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE6: [PMC] This should be programmed to RX power down mode enable for DQS
17	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE6: [PMC] This should be programmed to TX power down mode enable for DQS
16	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE6: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
15	0x1	DDLL_BIT_RXDQ_PWRD_BYTE5: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
14	0x0	DDLL_BIT_TXDQ_PWRD_BYTE5: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
12	0x1	DDLL_BYTE_RXRT_PWRD_BYTE5: [PMC] This should be programmed to RX power down mode enable for retiming flop
11	0x0	DDLL_BYTE_QU_PWRD_BYTE5: [PMC] This should be programmed to QUSE power down mode enable for
10	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE5: [PMC] This should be programmed to RX power down mode enable for DQS
9	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE5: [PMC] This should be programmed to TX power down mode enable for DQS
8	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE5: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
7	0x1	DDLL_BIT_RXDQ_PWRD_BYTE4: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
6	0x0	DDLL_BIT_TXDQ_PWRD_BYTE4: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
4	0x1	DDLL_BYTE_RXRT_PWRD_BYTE4: [PMC] This should be programmed to RX power down mode enable for retiming flop
3	0x0	DDLL_BYTE_QU_PWRD_BYTE4: [PMC] This should be programmed to QUSE power down mode enable for
2	0x1	DDLL_BYTE_RXDQS_PWRD_BYTE4: [PMC] This should be programmed to RX power down mode enable for DQS
1	0x0	DDLL_BYTE_TXDQS_PWRD_BYTE4: [PMC] This should be programmed to TX power down mode enable for DQS
0	0x0	DDLL_BYTE_TXDQ_PWRD_BYTE4: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping

18.11.2.288 EMC_PMACRO_DDLL_PWRD_2_0

DLL power down mode enables

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x778 | Read/Write: R/W | Reset: 0x94949494 (0b10x1010010x1010010x1010010x10100)

Bit	Reset	Description
31	0x1	DDLL_BIT_RXDQ_PWRD_CMD3: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
30	0x0	DDLL_BIT_TXDQ_PWRD_CMD3: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
28	0x1	DDLL_BYTE_RXRT_PWRD_CMD3: [PMC] This should be programmed to RX power down mode enable for retiming flop
27	0x0	DDLL_BYTE_QU_PWRD_CMD3: [PMC] This should be programmed to QUSE power down mode enable for
26	0x1	DDLL_BYTE_RXDQS_PWRD_CMD3: [PMC] This should be programmed to RX power down mode enable for DQS
25	0x0	DDLL_BYTE_TXDQS_PWRD_CMD3: [PMC] This should be programmed to TX power down mode enable for DQS
24	0x0	DDLL_BYTE_TXDQ_PWRD_CMD3: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
23	0x1	DDLL_BIT_RXDQ_PWRD_CMD2: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
22	0x0	DDLL_BIT_TXDQ_PWRD_CMD2: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
20	0x1	DDLL_BYTE_RXRT_PWRD_CMD2: [PMC] This should be programmed to RX power down mode enable for retiming flop
19	0x0	DDLL_BYTE_QU_PWRD_CMD2: [PMC] This should be programmed to QUSE power down mode enable for
18	0x1	DDLL_BYTE_RXDQS_PWRD_CMD2: [PMC] This should be programmed to RX power down mode enable for DQS
17	0x0	DDLL_BYTE_TXDQS_PWRD_CMD2: [PMC] This should be programmed to TX power down mode enable for DQS
16	0x0	DDLL_BYTE_TXDQ_PWRD_CMD2: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
15	0x1	DDLL_BIT_RXDQ_PWRD_CMD1: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
14	0x0	DDLL_BIT_TXDQ_PWRD_CMD1: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
12	0x1	DDLL_BYTE_RXRT_PWRD_CMD1: [PMC] This should be programmed to RX power down mode enable for retiming flop
11	0x0	DDLL_BYTE_QU_PWRD_CMD1: [PMC] This should be programmed to QUSE power down mode enable for
10	0x1	DDLL_BYTE_RXDQS_PWRD_CMD1: [PMC] This should be programmed to RX power down mode enable for DQS
9	0x0	DDLL_BYTE_TXDQS_PWRD_CMD1: [PMC] This should be programmed to TX power down mode enable for DQS
8	0x0	DDLL_BYTE_TXDQ_PWRD_CMD1: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping
7	0x1	DDLL_BIT_RXDQ_PWRD_CMD0: [PMC] This should be programmed to RX fine trimmer power down enable value for DQS
6	0x0	DDLL_BIT_TXDQ_PWRD_CMD0: [PMC] This should be programmed to TX fine trimmer power down enable value for DQ
4	0x1	DDLL_BYTE_RXRT_PWRD_CMD0: [PMC] This should be programmed to RX power down mode enable for retiming flop
3	0x0	DDLL_BYTE_QU_PWRD_CMD0: [PMC] This should be programmed to QUSE power down mode enable for
2	0x1	DDLL_BYTE_RXDQS_PWRD_CMD0: [PMC] This should be programmed to RX power down mode enable for DQS
1	0x0	DDLL_BYTE_TXDQS_PWRD_CMD0: [PMC] This should be programmed to TX power down mode enable for DQS
0	0x0	DDLL_BYTE_TXDQ_PWRD_CMD0: [PMC] This should be programmed to TX power down mode enable for DQ. 1= power down mode, output clamped, 0= normal mode without clamping

18.11.2.289 EMC_PMACRO_CMD_CTRL_0_0

DLL bypass and power down mode enables for CMD I/O

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x780 | Read/Write: R/W | Reset: 0x0a0a0a0a (0bxx001010xx001010xx001010xx001010)

Bit	Reset	Description
29	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE3: [PMC] Bypass the DDLL calibration value
28	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE3: [PMC] Bypass the DDLL calibration value
27	0x1	DDLL_BIT_TXCMD_BYPASS_CKE3: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
26	0x0	DDLL_BYTE_TXCMD_PWRD_CKE3: [PMC] This should be programmed to TX power down mode enable for CMD
25	0x1	DDLL_BYT_TXCMD_BYPASS_CKE3: [PMC] This should be programmed to TX trimmer bypass value for CMD
24	0x0	CMD_E_INPUT_CKE3: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
21	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE2: [PMC] Bypass the DDLL calibration value
20	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE2: [PMC] Bypass the DDLL calibration value
19	0x1	DDLL_BIT_TXCMD_BYPASS_CKE2: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
18	0x0	DDLL_BYTE_TXCMD_PWRD_CKE2: [PMC] This should be programmed to TX power down mode enable for CMD
17	0x1	DDLL_BYT_TXCMD_BYPASS_CKE2: [PMC] This should be programmed to TX trimmer bypass value for CMD
16	0x0	CMD_E_INPUT_CKE2: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
13	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE1: [PMC] Bypass the DDLL calibration value
12	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE1: [PMC] Bypass the DDLL calibration value
11	0x1	DDLL_BIT_TXCMD_BYPASS_CKE1: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
10	0x0	DDLL_BYTE_TXCMD_PWRD_CKE1: [PMC] This should be programmed to TX power down mode enable for CMD
9	0x1	DDLL_BYT_TXCMD_BYPASS_CKE1: [PMC] This should be programmed to TX trimmer bypass value for CMD
8	0x0	CMD_E_INPUT_CKE1: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
5	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE0: [PMC] Bypass the DDLL calibration value
4	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE0: [PMC] Bypass the DDLL calibration value
3	0x1	DDLL_BIT_TXCMD_BYPASS_CKE0: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
2	0x0	DDLL_BYTE_TXCMD_PWRD_CKE0: [PMC] This should be programmed to TX power down mode enable for CMD
1	0x1	DDLL_BYTE_TXCMD_BYPASS_CKE0: [PMC] This should be programmed to TX trimmer bypass value for CMD
0	0x0	CMD_E_INPUT_CKE0: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted

18.11.2.290 EMC_PMACRO_CMD_CTRL_1_0

DLL bypass and power down mode enables for CMD I/O

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x784 | Read/Write: R/W | Reset: 0x0a0a0a0a (0bxx001010xx001010xx001010xx001010)

Bit	Reset	Description
29	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE7: [PMC] Bypass the DDLL calibration value
28	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE7: [PMC] Bypass the DDLL calibration value
27	0x1	DDLL_BIT_TXCMD_BYPASS_CKE7: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
26	0x0	DDLL_BYTE_TXCMD_PWRD_CKE7: [PMC] This should be programmed to TX power down mode enable for CMD
25	0x1	DDLL_BYT_TXCMD_BYPASS_CKE7: [PMC] This should be programmed to TX trimmer bypass value for CMD
24	0x0	CMD_E_INPUT_CKE7: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
21	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE6: [PMC] Bypass the DDLL calibration value
20	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE6: [PMC] Bypass the DDLL calibration value
19	0x1	DDLL_BIT_TXCMD_BYPASS_CKE6: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
18	0x0	DDLL_BYTE_TXCMD_PWRD_CKE6: [PMC] This should be programmed to TX power down mode enable for CMD
17	0x1	DDLL_BYT_TXCMD_BYPASS_CKE6: [PMC] This should be programmed to TX trimmer bypass value for CMD
16	0x0	CMD_E_INPUT_CKE6: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
13	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE5: [PMC] Bypass the DDLL calibration value
12	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE5: [PMC] Bypass the DDLL calibration value
11	0x1	DDLL_BIT_TXCMD_BYPASS_CKE5: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
10	0x0	DDLL_BYTE_TXCMD_PWRD_CKE5: [PMC] This should be programmed to TX power down mode enable for CMD
9	0x1	DDLL_BYT_TXCMD_BYPASS_CKE5: [PMC] This should be programmed to TX trimmer bypass value for CMD
8	0x0	CMD_E_INPUT_CKE5: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
5	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_CKE4: [PMC] Bypass the DDLL calibration value
4	0x0	DDLLCAL_BIT_TXCMD_BYPASS_CKE4: [PMC] Bypass the DDLL calibration value
3	0x1	DDLL_BIT_TXCMD_BYPASS_CKE4: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
2	0x0	DDLL_BYTE_TXCMD_PWRD_CKE4: [PMC] This should be programmed to TX power down mode enable for CMD
1	0x1	DDLL_BYT_TXCMD_BYPASS_CKE4: [PMC] This should be programmed to TX trimmer bypass value for CMD
0	0x0	CMD_E_INPUT_CKE4: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted

18.11.2.291 EMC_PMACRO_CMD_CTRL_2_0

DLL bypass and power down mode enables for CMD I/O

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x788 | Read/Write: R/W | Reset: 0x0a0a0a0a (0bxx001010xx001010xx001010xx001010)

Bit	Reset	Description
29	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_MISC2: [PMC] Bypass the DDLL calibration value
28	0x0	DDLLCAL_BIT_TXCMD_BYPASS_MISC2: [PMC] Bypass the DDLL calibration value
27	0x1	DDLL_BIT_TXCMD_BYPASS_MISC2: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
26	0x0	DDLL_BYTE_TXCMD_PWRD_MISC2: [PMC] This should be programmed to TX power down mode enable for CMD
25	0x1	DDLL_BYT_TXCMD_BYPASS_MISC2: [PMC] This should be programmed to TX trimmer bypass value for CMD
24	0x0	CMD_E_INPUT_MISC2: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
21	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_MISC1: [PMC] Bypass the DDLL calibration value
20	0x0	DDLLCAL_BIT_TXCMD_BYPASS_MISC1: [PMC] Bypass the DDLL calibration value
19	0x1	DDLL_BIT_TXCMD_BYPASS_MISC1: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
18	0x0	DDLL_BYTE_TXCMD_PWRD_MISC1: [PMC] This should be programmed to TX power down mode enable for CMD
17	0x1	DDLL_BYT_TXCMD_BYPASS_MISC1: [PMC] This should be programmed to TX trimmer bypass value for CMD
16	0x0	CMD_E_INPUT_MISC1: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
13	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_MISC0: [PMC] Bypass the DDLL calibration value
12	0x0	DDLLCAL_BIT_TXCMD_BYPASS_MISC0: [PMC] Bypass the DDLL calibration value
11	0x1	DDLL_BIT_TXCMD_BYPASS_MISC0: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
10	0x0	DDLL_BYTE_TXCMD_PWRD_MISC0: [PMC] This should be programmed to TX power down mode enable for CMD
9	0x1	DDLL_BYT_TXCMD_BYPASS_MISC0: [PMC] This should be programmed to TX trimmer bypass value for CMD
8	0x0	CMD_E_INPUT_MISC0: Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted
5	0x0	DDLLCAL_BYTE_TXCMD_BYPASS_RESET: [PMC] Bypass the DDLL calibration value
4	0x0	DDLLCAL_BIT_TXCMD_BYPASS_RESET: [PMC] Bypass the DDLL calibration value
3	0x1	DDLL_BIT_TXCMD_BYPASS_RESET: [PMC] This should be programmed to TX fine trimmer bypass value for CMD
2	0x0	DDLL_BYTE_TXCMD_PWRD_RESET: [PMC] This should be programmed to TX power down mode enable for CMD
1	0x1	DDLL_BYT_TXCMD_BYPASS_RESET: [PMC] This should be programmed to TX trimmer bypass value for CMD
0	0x0	CMD_E_INPUT_RESET: [PMC] Enable input path for IO_CMD; Latency =8ns, time for receiver to become fully functional after E_INPUT asserted

18.11.2.292 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE0_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x800 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE0: This should be programmed to TX fine trimmer value for pin3 of BYTE0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE0: This should be programmed to TX fine trimmer value for pin2 of BYTE0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE0: This should be programmed to TX fine trimmer value for pin1 of BYTE0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE0: This should be programmed to TX fine trimmer value for pin0 of BYTE0

18.11.2.293 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE0_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x804 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE0: This should be programmed to TX fine trimmer value for pin7 of BYTE0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE0: This should be programmed to TX fine trimmer value for pin6 of BYTE0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE0: This should be programmed to TX fine trimmer value for pin5 of BYTE0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE0: This should be programmed to TX fine trimmer value for pin4 of BYTE0

18.11.2.294 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE0_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x808 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE0: This should be programmed to TX fine trimmer value for pin8 of BYTE0

18.11.2.295 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE0_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x80c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE0: This should be programmed to TX fine trimmer value for pin10 of BYTE0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE0: This should be programmed to TX fine trimmer value for pin9 of BYTE0

18.11.2.296 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE1_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x810 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE1: This should be programmed to TX fine trimmer value for pin3 of BYTE1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE1: This should be programmed to TX fine trimmer value for pin2 of BYTE1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE1: This should be programmed to TX fine trimmer value for pin1 of BYTE1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE1: This should be programmed to TX fine trimmer value for pin0 of BYTE1

18.11.2.297 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE1_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x814 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE1: This should be programmed to TX fine trimmer value for pin7 of BYTE1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE1: This should be programmed to TX fine trimmer value for pin6 of BYTE1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE1: This should be programmed to TX fine trimmer value for pin5 of BYTE1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE1: This should be programmed to TX fine trimmer value for pin4 of BYTE1

18.11.2.298 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE1_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x818 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE1: This should be programmed to TX fine trimmer value for pin8 of BYTE1

18.11.2.299 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE1_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x81c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE1: This should be programmed to TX fine trimmer value for pin10 of BYTE1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE1: This should be programmed to TX fine trimmer value for pin9 of BYTE1

18.11.2.300 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE2_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x820 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE2: This should be programmed to TX fine trimmer value for pin3 of BYTE2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE2: This should be programmed to TX fine trimmer value for pin2 of BYTE2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE2: This should be programmed to TX fine trimmer value for pin1 of BYTE2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE2: This should be programmed to TX fine trimmer value for pin0 of BYTE2

18.11.2.301 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE2_1_0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x824 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE2: This should be programmed to TX fine trimmer value for pin7 of BYTE2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE2: This should be programmed to TX fine trimmer value for pin6 of BYTE2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE2: This should be programmed to TX fine trimmer value for pin5 of BYTE2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE2: This should be programmed to TX fine trimmer value for pin4 of BYTE2

18.11.2.302 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE2_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x828 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE2: This should be programmed to TX fine trimmer value for pin8 of BYTE2

18.11.2.303 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE2_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x82c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE2: This should be programmed to TX fine trimmer value for pin10 of BYTE2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE2: This should be programmed to TX fine trimmer value for pin9 of BYTE2

18.11.2.304 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE3_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x830 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE3: This should be programmed to TX fine trimmer value for pin3 of BYTE3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE3: This should be programmed to TX fine trimmer value for pin2 of BYTE3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE3: This should be programmed to TX fine trimmer value for pin1 of BYTE3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE3: This should be programmed to TX fine trimmer value for pin0 of BYTE3

18.11.2.305 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE3_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x834 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE3: This should be programmed to TX fine trimmer value for pin7 of BYTE3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE3: This should be programmed to TX fine trimmer value for pin6 of BYTE3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE3: This should be programmed to TX fine trimmer value for pin5 of BYTE3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE3: This should be programmed to TX fine trimmer value for pin4 of BYTE3

18.11.2.306 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE3_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x838 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE3: This should be programmed to TX fine trimmer value for pin8 of BYTE3

18.11.2.307 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE3_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x83c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE3: This should be programmed to TX fine trimmer value for pin10 of BYTE3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE3: This should be programmed to TX fine trimmer value for pin9 of BYTE3

18.11.2.308 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE4_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x840 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE4: This should be programmed to TX fine trimmer value for pin3 of BYTE4
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE4: This should be programmed to TX fine trimmer value for pin2 of BYTE4
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE4: This should be programmed to TX fine trimmer value for pin1 of BYTE4
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE4: This should be programmed to TX fine trimmer value for pin0 of BYTE4

18.11.2.309 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE4_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x844 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE4: This should be programmed to TX fine trimmer value for pin7 of BYTE4
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE4: This should be programmed to TX fine trimmer value for pin6 of BYTE4
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE4: This should be programmed to TX fine trimmer value for pin5 of BYTE4
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE4: This should be programmed to TX fine trimmer value for pin4 of BYTE4

18.11.2.310 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE4_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x848 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE4: This should be programmed to TX fine trimmer value for pin8 of BYTE4

18.11.2.311 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE4_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x84c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE4: This should be programmed to TX fine trimmer value for pin10 of BYTE4
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE4: This should be programmed to TX fine trimmer value for pin9 of BYTE4

18.11.2.312 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE5_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x850 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE5: This should be programmed to TX fine trimmer value for pin3 of BYTE5
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE5: This should be programmed to TX fine trimmer value for pin2 of BYTE5
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE5: This should be programmed to TX fine trimmer value for pin1 of BYTE5
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE5: This should be programmed to TX fine trimmer value for pin0 of BYTE5

18.11.2.313 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE5_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0x854 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE5: This should be programmed to TX fine trimmer value for pin7 of BYTE5
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE5: This should be programmed to TX fine trimmer value for pin6 of BYTE5
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE5: This should be programmed to TX fine trimmer value for pin5 of BYTE5
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE5: This should be programmed to TX fine trimmer value for pin4 of BYTE5

18.11.2.314 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE5_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE5: This should be programmed to TX fine trimmer value for pin8 of BYTE5

18.11.2.315 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE5_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x85c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE5: This should be programmed to TX fine trimmer value for pin10 of BYTE5
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE5: This should be programmed to TX fine trimmer value for pin9 of BYTE5

18.11.2.316 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE6_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x860 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE6: This should be programmed to TX fine trimmer value for pin3 of BYTE6
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE6: This should be programmed to TX fine trimmer value for pin2 of BYTE6
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE6: This should be programmed to TX fine trimmer value for pin1 of BYTE6
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE6: This should be programmed to TX fine trimmer value for pin0 of BYTE6

18.11.2.317 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE6_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x864 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE6: This should be programmed to TX fine trimmer value for pin7 of BYTE6
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE6: This should be programmed to TX fine trimmer value for pin6 of BYTE6
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE6: This should be programmed to TX fine trimmer value for pin5 of BYTE6
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE6: This should be programmed to TX fine trimmer value for pin4 of BYTE6

18.11.2.318 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE6_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x868 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE6: This should be programmed to TX fine trimmer value for pin8 of BYTE6

18.11.2.319 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE6_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x86c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE6: This should be programmed to TX fine trimmer value for pin10 of BYTE6
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE6: This should be programmed to TX fine trimmer value for pin9 of BYTE6

18.11.2.320 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE7_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x870 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE7: This should be programmed to TX fine trimmer value for pin3 of BYTE7
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE7: This should be programmed to TX fine trimmer value for pin2 of BYTE7
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE7: This should be programmed to TX fine trimmer value for pin1 of BYTE7
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE7: This should be programmed to TX fine trimmer value for pin0 of BYTE7

18.11.2.321 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE7_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x874 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE7: This should be programmed to TX fine trimmer value for pin7 of BYTE7
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE7: This should be programmed to TX fine trimmer value for pin6 of BYTE7
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE7: This should be programmed to TX fine trimmer value for pin5 of BYTE7
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE7: This should be programmed to TX fine trimmer value for pin4 of BYTE7

18.11.2.322 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE7_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x878 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE7: This should be programmed to TX fine trimmer value for pin8 of BYTE7

18.11.2.323 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_BYTE7_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x87c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_BYTE7: This should be programmed to TX fine trimmer value for pin10 of BYTE7
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_BYTE7: This should be programmed to TX fine trimmer value for pin9 of BYTE7

18.11.2.324 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD0_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x880 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_CMD0: This should be programmed to TX fine trimmer value for pin3 of CMD0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_CMD0: This should be programmed to TX fine trimmer value for pin2 of CMD0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_CMD0: This should be programmed to TX fine trimmer value for pin1 of CMD0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_CMD0: This should be programmed to TX fine trimmer value for pin0 of CMD0

18.11.2.325 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD0_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x884 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_CMD0: This should be programmed to TX fine trimmer value for pin7 of CMD0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_CMD0: This should be programmed to TX fine trimmer value for pin6 of CMD0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_CMD0: This should be programmed to TX fine trimmer value for pin5 of CMD0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_CMD0: This should be programmed to TX fine trimmer value for pin4 of CMD0

18.11.2.326 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD0_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x888 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_CMD0: This should be programmed to TX fine trimmer value for pin8 of CMD0

18.11.2.327 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD0_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x88c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_CMD0: This should be programmed to TX fine trimmer value for pin10 of CMD0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_CMD0: This should be programmed to TX fine trimmer value for pin9 of CMD0

18.11.2.328 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD1_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x890 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_CMD1: This should be programmed to TX fine trimmer value for pin3 of CMD1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_CMD1: This should be programmed to TX fine trimmer value for pin2 of CMD1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_CMD1: This should be programmed to TX fine trimmer value for pin1 of CMD1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_CMD1: This should be programmed to TX fine trimmer value for pin0 of CMD1

18.11.2.329 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD1_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x894 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_CMD1: This should be programmed to TX fine trimmer value for pin7 of CMD1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_CMD1: This should be programmed to TX fine trimmer value for pin6 of CMD1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_CMD1: This should be programmed to TX fine trimmer value for pin5 of CMD1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_CMD1: This should be programmed to TX fine trimmer value for pin4 of CMD1

18.11.2.330 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD1_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x898 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_CMD1: This should be programmed to TX fine trimmer value for pin8 of CMD1

18.11.2.331 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD1_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x89c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_CMD1: This should be programmed to TX fine trimmer value for pin10 of CMD1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_CMD1: This should be programmed to TX fine trimmer value for pin9 of CMD1

18.11.2.332 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD2_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x8a0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_CMD2: This should be programmed to TX fine trimmer value for pin3 of CMD2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_CMD2: This should be programmed to TX fine trimmer value for pin2 of CMD2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_CMD2: This should be programmed to TX fine trimmer value for pin1 of CMD2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_CMD2: This should be programmed to TX fine trimmer value for pin0 of CMD2

18.11.2.333 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD2_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x8a4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_CMD2: This should be programmed to TX fine trimmer value for pin7 of CMD2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_CMD2: This should be programmed to TX fine trimmer value for pin6 of CMD2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_CMD2: This should be programmed to TX fine trimmer value for pin5 of CMD2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_CMD2: This should be programmed to TX fine trimmer value for pin4 of CMD2

18.11.2.334 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD2_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x8a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_CMD2: This should be programmed to TX fine trimmer value for pin8 of CMD2

18.11.2.335 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD2_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x8ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_CMD2: This should be programmed to TX fine trimmer value for pin10 of CMD2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_CMD2: This should be programmed to TX fine trimmer value for pin9 of CMD2

18.11.2.336 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD3_0_0

Rank0 OB SHORT DQ DDLL value for pin0 to pin3 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x8b0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN3_CMD3: This should be programmed to TX fine trimmer value for pin3 of CMD3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN2_CMD3: This should be programmed to TX fine trimmer value for pin2 of CMD3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN1_CMD3: This should be programmed to TX fine trimmer value for pin1 of CMD3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN0_CMD3: This should be programmed to TX fine trimmer value for pin0 of CMD3

18.11.2.337 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD3_1_0

Rank0 OB SHORT DQ DDLL value for pin4 to pin7 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0x8b4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN7_CMD3: This should be programmed to TX fine trimmer value for pin7 of CMD3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN6_CMD3: This should be programmed to TX fine trimmer value for pin6 of CMD3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN5_CMD3: This should be programmed to TX fine trimmer value for pin5 of CMD3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN4_CMD3: This should be programmed to TX fine trimmer value for pin4 of CMD3

18.11.2.338 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD3_2_0

Rank0 OB SHORT DQ DDLL value for pin8 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x8b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN8_CMD3: This should be programmed to TX fine trimmer value for pin8 of CMD3

18.11.2.339 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK0_CMD3_3_0

Rank0 OB SHORT DQ DDLL value for pin9 to pin10 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x8bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN10_CMD3: This should be programmed to TX fine trimmer value for pin10 of CMD3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK0_PIN9_CMD3: This should be programmed to TX fine trimmer value for pin9 of CMD3

18.11.2.340 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE0_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin3 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x900 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE0: This should be programmed to TX fine trimmer value for pin3 of BYTE0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE0: This should be programmed to TX fine trimmer value for pin2 of BYTE0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE0: This should be programmed to TX fine trimmer value for pin1 of BYTE0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE0: This should be programmed to TX fine trimmer value for pin0 of BYTE0

18.11.2.341 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE0_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x904 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE0: This should be programmed to TX fine trimmer value for pin7 of BYTE0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE0: This should be programmed to TX fine trimmer value for pin6 of BYTE0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE0: This should be programmed to TX fine trimmer value for pin5 of BYTE0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE0: This should be programmed to TX fine trimmer value for pin4 of BYTE0

18.11.2.342 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE0_2_0

Rank1 OB SHORT DQ DDLL value for pin8 to pin10 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x908 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE0: This should be programmed to TX fine trimmer value for pin8 of BYTE0

18.11.2.343 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE0_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x90c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE0: This should be programmed to TX fine trimmer value for pin10 of BYTE0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE0: This should be programmed to TX fine trimmer value for pin9 of BYTE0

18.11.2.344 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE1_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin3 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x910 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE1: This should be programmed to TX fine trimmer value for pin3 of BYTE1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE1: This should be programmed to TX fine trimmer value for pin2 of BYTE1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE1: This should be programmed to TX fine trimmer value for pin1 of BYTE1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE1: This should be programmed to TX fine trimmer value for pin0 of BYTE1

18.11.2.345 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE1_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x914 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE1: This should be programmed to TX fine trimmer value for pin7 of BYTE1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE1: This should be programmed to TX fine trimmer value for pin6 of BYTE1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE1: This should be programmed to TX fine trimmer value for pin5 of BYTE1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE1: This should be programmed to TX fine trimmer value for pin4 of BYTE1

18.11.2.346 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE1_2_0

Rank1 OB SHORT DQ DDLL value for pin8 to pin10 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x918 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE1: This should be programmed to TX fine trimmer value for pin8 of BYTE1

18.11.2.347 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE1_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x91c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE1: This should be programmed to TX fine trimmer value for pin10 of BYTE1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE1: This should be programmed to TX fine trimmer value for pin9 of BYTE1

18.11.2.348 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE2_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin3 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x920 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE2: This should be programmed to TX fine trimmer value for pin3 of BYTE2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE2: This should be programmed to TX fine trimmer value for pin2 of BYTE2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE2: This should be programmed to TX fine trimmer value for pin1 of BYTE2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE2: This should be programmed to TX fine trimmer value for pin0 of BYTE2

18.11.2.349 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE2_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x924 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE2: This should be programmed to TX fine trimmer value for pin7 of BYTE2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE2: This should be programmed to TX fine trimmer value for pin6 of BYTE2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE2: This should be programmed to TX fine trimmer value for pin5 of BYTE2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE2: This should be programmed to TX fine trimmer value for pin4 of BYTE2

18.11.2.350 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE2_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x928 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE2: This should be programmed to TX fine trimmer value for pin8 of BYTE2

18.11.2.351 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE2_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x92c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE2: This should be programmed to TX fine trimmer value for pin10 of BYTE2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE2: This should be programmed to TX fine trimmer value for pin9 of BYTE2

18.11.2.352 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE3_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin3 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x930 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE3: This should be programmed to TX fine trimmer value for pin3 of BYTE3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE3: This should be programmed to TX fine trimmer value for pin2 of BYTE3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE3: This should be programmed to TX fine trimmer value for pin1 of BYTE3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE3: This should be programmed to TX fine trimmer value for pin0 of BYTE3

18.11.2.353 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE3_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x934 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE3: This should be programmed to TX fine trimmer value for pin7 of BYTE3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE3: This should be programmed to TX fine trimmer value for pin6 of BYTE3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE3: This should be programmed to TX fine trimmer value for pin5 of BYTE3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE3: This should be programmed to TX fine trimmer value for pin4 of BYTE3

18.11.2.354 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE3_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x938 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE3: This should be programmed to TX fine trimmer value for pin8 of BYTE3

18.11.2.355 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE3_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x93c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE3: This should be programmed to TX fine trimmer value for pin10 of BYTE3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE3: This should be programmed to TX fine trimmer value for pin9 of BYTE3

18.11.2.356 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE4_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin3 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0x940 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE4: This should be programmed to TX fine trimmer value for pin3 of BYTE4
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE4: This should be programmed to TX fine trimmer value for pin2 of BYTE4
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE4: This should be programmed to TX fine trimmer value for pin1 of BYTE4
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE4: This should be programmed to TX fine trimmer value for pin0 of BYTE4

18.11.2.357 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE4_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x944 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE4: This should be programmed to TX fine trimmer value for pin7 of BYTE4
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE4: This should be programmed to TX fine trimmer value for pin6 of BYTE4
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE4: This should be programmed to TX fine trimmer value for pin5 of BYTE4
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE4: This should be programmed to TX fine trimmer value for pin4 of BYTE4

18.11.2.358 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE4_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x948 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE4: This should be programmed to TX fine trimmer value for pin8 of BYTE4

18.11.2.359 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE4_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x94c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE4: This should be programmed to TX fine trimmer value for pin10 of BYTE4
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE4: This should be programmed to TX fine trimmer value for pin9 of BYTE4

18.11.2.360 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE5_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin3 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x950 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE5: This should be programmed to TX fine trimmer value for pin3 of BYTE5
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE5: This should be programmed to TX fine trimmer value for pin2 of BYTE5
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE5: This should be programmed to TX fine trimmer value for pin1 of BYTE5
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE5: This should be programmed to TX fine trimmer value for pin0 of BYTE5

18.11.2.361 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE5_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x954 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE5: This should be programmed to TX fine trimmer value for pin7 of BYTE5
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE5: This should be programmed to TX fine trimmer value for pin6 of BYTE5
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE5: This should be programmed to TX fine trimmer value for pin5 of BYTE5
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE5: This should be programmed to TX fine trimmer value for pin4 of BYTE5

18.11.2.362 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE5_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x958 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE5: This should be programmed to TX fine trimmer value for pin8 of BYTE5

18.11.2.363 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE5_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x95c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE5: This should be programmed to TX fine trimmer value for pin10 of BYTE5
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE5: This should be programmed to TX fine trimmer value for pin9 of BYTE5

18.11.2.364 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE6_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin1 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0x960 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE6: This should be programmed to TX fine trimmer value for pin3 of BYTE6
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE6: This should be programmed to TX fine trimmer value for pin2 of BYTE6
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE6: This should be programmed to TX fine trimmer value for pin1 of BYTE6
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE6: This should be programmed to TX fine trimmer value for pin0 of BYTE6

18.11.2.365 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE6_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x964 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE6: This should be programmed to TX fine trimmer value for pin7 of BYTE6
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE6: This should be programmed to TX fine trimmer value for pin6 of BYTE6
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE6: This should be programmed to TX fine trimmer value for pin5 of BYTE6
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE6: This should be programmed to TX fine trimmer value for pin4 of BYTE6

18.11.2.366 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE6_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x968 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE6: This should be programmed to TX fine trimmer value for pin8 of BYTE6

18.11.2.367 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE6_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x96c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE6: This should be programmed to TX fine trimmer value for pin10 of BYTE6
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE6: This should be programmed to TX fine trimmer value for pin9 of BYTE6

18.11.2.368 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE7_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin1 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x970 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE7: This should be programmed to TX fine trimmer value for pin3 of BYTE7
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE7: This should be programmed to TX fine trimmer value for pin2 of BYTE7
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE7: This should be programmed to TX fine trimmer value for pin1 of BYTE7
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE7: This should be programmed to TX fine trimmer value for pin0 of BYTE7

18.11.2.369 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE7_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x974 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE7: This should be programmed to TX fine trimmer value for pin7 of BYTE7
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE7: This should be programmed to TX fine trimmer value for pin6 of BYTE7
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE7: This should be programmed to TX fine trimmer value for pin5 of BYTE7
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE7: This should be programmed to TX fine trimmer value for pin4 of BYTE7

18.11.2.370 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE7_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0x978 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE7: This should be programmed to TX fine trimmer value for pin8 of BYTE7

18.11.2.371 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_BYTE7_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x97c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_BYTE7: This should be programmed to TX fine trimmer value for pin10 of BYTE7
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_BYTE7: This should be programmed to TX fine trimmer value for pin9 of BYTE7

18.11.2.372 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD0_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin1 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x980 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_CMD0: This should be programmed to TX fine trimmer value for pin3 of CMD0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_CMD0: This should be programmed to TX fine trimmer value for pin2 of CMD0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_CMD0: This should be programmed to TX fine trimmer value for pin1 of CMD0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_CMD0: This should be programmed to TX fine trimmer value for pin0 of CMD0

18.11.2.373 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD0_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x984 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_CMD0: This should be programmed to TX fine trimmer value for pin7 of CMD0
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_CMD0: This should be programmed to TX fine trimmer value for pin6 of CMD0
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_CMD0: This should be programmed to TX fine trimmer value for pin5 of CMD0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_CMD0: This should be programmed to TX fine trimmer value for pin4 of CMD0

18.11.2.374 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD0_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x988 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_CMD0: This should be programmed to TX fine trimmer value for pin8 of CMD0

18.11.2.375 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD0_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x98c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_CMD0: This should be programmed to TX fine trimmer value for pin10 of CMD0
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_CMD0: This should be programmed to TX fine trimmer value for pin9 of CMD0

18.11.2.376 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD1_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin1 of cmd1 brick.

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x990 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_CMD1: This should be programmed to TX fine trimmer value for pin3 of CMD1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_CMD1: This should be programmed to TX fine trimmer value for pin2 of CMD1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_CMD1: This should be programmed to TX fine trimmer value for pin1 of CMD1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_CMD1: This should be programmed to TX fine trimmer value for pin0 of CMD1

18.11.2.377 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD1_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x994 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_CMD1: This should be programmed to TX fine trimmer value for pin7 of CMD1
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_CMD1: This should be programmed to TX fine trimmer value for pin6 of CMD1
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_CMD1: This should be programmed to TX fine trimmer value for pin5 of CMD1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_CMD1: This should be programmed to TX fine trimmer value for pin4 of CMD1

18.11.2.378 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD1_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x998 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_CMD1: This should be programmed to TX fine trimmer value for pin8 of CMD1

18.11.2.379 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD1_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x99c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_CMD1: This should be programmed to TX fine trimmer value for pin10 of CMD1
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_CMD1: This should be programmed to TX fine trimmer value for pin9 of CMD1

18.11.2.380 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD2_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin1 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9a0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_CMD2: This should be programmed to TX fine trimmer value for pin3 of CMD2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_CMD2: This should be programmed to TX fine trimmer value for pin2 of CMD2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_CMD2: This should be programmed to TX fine trimmer value for pin1 of CMD2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_CMD2: This should be programmed to TX fine trimmer value for pin0 of CMD2

18.11.2.381 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD2_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9a4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_CMD2: This should be programmed to TX fine trimmer value for pin7 of CMD2
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_CMD2: This should be programmed to TX fine trimmer value for pin6 of CMD2
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_CMD2: This should be programmed to TX fine trimmer value for pin5 of CMD2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_CMD2: This should be programmed to TX fine trimmer value for pin4 of CMD2

18.11.2.382 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD2_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_CMD2: This should be programmed to TX fine trimmer value for pin8 of CMD2

18.11.2.383 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD2_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_CMD2: This should be programmed to TX fine trimmer value for pin10 of CMD2
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_CMD2: This should be programmed to TX fine trimmer value for pin9 of CMD2

18.11.2.384 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD3_0_0

Rank1 OB SHORT DQ DDLL value for pin0 to pin1 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9b0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN3_CMD3: This should be programmed to TX fine trimmer value for pin3 of CMD3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN2_CMD3: This should be programmed to TX fine trimmer value for pin2 of CMD3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN1_CMD3: This should be programmed to TX fine trimmer value for pin1 of CMD3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN0_CMD3: This should be programmed to TX fine trimmer value for pin0 of CMD3

18.11.2.385 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD3_1_0

Rank1 OB SHORT DQ DDLL value for pin4 to pin7 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9b4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN7_CMD3: This should be programmed to TX fine trimmer value for pin7 of CMD3
22:16	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN6_CMD3: This should be programmed to TX fine trimmer value for pin6 of CMD3
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN5_CMD3: This should be programmed to TX fine trimmer value for pin5 of CMD3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN4_CMD3: This should be programmed to TX fine trimmer value for pin4 of CMD3

18.11.2.386 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD3_2_0

Rank1 OB SHORT DQ DDLL value for pin8 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN8_CMD3: This should be programmed to TX fine trimmer value for pin8 of CMD3

18.11.2.387 EMC_PMACRO_OB_DDLL_SHORT_DQ_RANK1_CMD3_3_0

Rank1 OB SHORT DQ DDLL value for pin9 to pin10 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0x9bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000x0000000)

Bit	Reset	Description
14:8	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN10_CMD3: This should be programmed to TX fine trimmer value for pin10 of CMD3
6:0	0x0	OB_DDLL_SHORT_DQ_RANK1_PIN9_CMD3: This should be programmed to TX fine trimmer value for pin9 of CMD3

18.11.2.388 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE0_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa00 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE0: This should be programmed to RX fine trimmer value for pin3 of BYTE0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE0: This should be programmed to RX fine trimmer value for pin2 of BYTE0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE0: This should be programmed to RX fine trimmer value for pin1 of BYTE0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE0: This should be programmed to RX fine trimmer value for pin0 of BYTE0

18.11.2.389 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE0_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa04 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE0: This should be programmed to RX fine trimmer value for pin7 of BYTE0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE0: This should be programmed to RX fine trimmer value for pin6 of BYTE0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE0: This should be programmed to RX fine trimmer value for pin5 of BYTE0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE0: This should be programmed to RX fine trimmer value for pin4 of BYTE0

18.11.2.390 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE0_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE0: This should be programmed to RX fine trimmer value for pin8 of BYTE0

18.11.2.391 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE1_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa10 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE1: This should be programmed to RX fine trimmer value for pin3 of BYTE1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE1: This should be programmed to RX fine trimmer value for pin2 of BYTE1

Bit	Reset	Description
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE1: This should be programmed to RX fine trimmer value for pin1 of BYTE1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE1: This should be programmed to RX fine trimmer value for pin0 of BYTE1

18.11.2.392 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE1_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa14 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE1: This should be programmed to RX fine trimmer value for pin7 of BYTE1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE1: This should be programmed to RX fine trimmer value for pin6 of BYTE1
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE1: This should be programmed to RX fine trimmer value for pin5 of BYTE1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE1: This should be programmed to RX fine trimmer value for pin4 of BYTE1

18.11.2.393 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE1_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE1: This should be programmed to RX fine trimmer value for pin8 of BYTE1

18.11.2.394 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE2_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa20 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE2: This should be programmed to RX fine trimmer value for pin3 of BYTE2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE2: This should be programmed to RX fine trimmer value for pin2 of BYTE2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE2: This should be programmed to RX fine trimmer value for pin1 of BYTE2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE2: This should be programmed to RX fine trimmer value for pin0 of BYTE2

18.11.2.395 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE2_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa24 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE2: This should be programmed to RX fine trimmer value for pin7 of BYTE2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE2: This should be programmed to RX fine trimmer value for pin6 of BYTE2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE2: This should be programmed to RX fine trimmer value for pin5 of BYTE2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE2: This should be programmed to RX fine trimmer value for pin4 of BYTE2

18.11.2.396 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE2_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE2: This should be programmed to RX fine trimmer value for pin8 of BYTE2

18.11.2.397 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE3_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa30 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE3: This should be programmed to RX fine trimmer value for pin3 of BYTE3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE3: This should be programmed to RX fine trimmer value for pin2 of BYTE3
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE3: This should be programmed to RX fine trimmer value for pin1 of BYTE3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE3: This should be programmed to RX fine trimmer value for pin0 of BYTE3

18.11.2.398 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE3_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa34 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE3: This should be programmed to RX fine trimmer value for pin7 of BYTE3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE3: This should be programmed to RX fine trimmer value for pin6 of BYTE3
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE3: This should be programmed to RX fine trimmer value for pin5 of BYTE3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE3: This should be programmed to RX fine trimmer value for pin4 of BYTE3

18.11.2.399 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE3_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE3: This should be programmed to RX fine trimmer value for pin8 of BYTE3

18.11.2.400 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE4_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin3 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa40 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE4: This should be programmed to RX fine trimmer value for pin3 of BYTE4
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE4: This should be programmed to RX fine trimmer value for pin2 of BYTE4
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE4: This should be programmed to RX fine trimmer value for pin1 of BYTE4
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE4: This should be programmed to RX fine trimmer value for pin0 of BYTE4

18.11.2.401 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE4_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa44 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE4: This should be programmed to RX fine trimmer value for pin7 of BYTE4
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE4: This should be programmed to RX fine trimmer value for pin6 of BYTE4
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE4: This should be programmed to RX fine trimmer value for pin5 of BYTE4
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE4: This should be programmed to RX fine trimmer value for pin4 of BYTE4

18.11.2.402 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE4_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE4: This should be programmed to RX fine trimmer value for pin8 of BYTE4

18.11.2.403 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE5_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin3 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa50 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE5: This should be programmed to RX fine trimmer value for pin3 of BYTE5
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE5: This should be programmed to RX fine trimmer value for pin2 of BYTE5
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE5: This should be programmed to RX fine trimmer value for pin1 of BYTE5
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE5: This should be programmed to RX fine trimmer value for pin0 of BYTE5

18.11.2.404 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE5_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa54 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE5: This should be programmed to RX fine trimmer value for pin7 of BYTE5
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE5: This should be programmed to RX fine trimmer value for pin6 of BYTE5
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE5: This should be programmed to RX fine trimmer value for pin5 of BYTE5
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE5: This should be programmed to RX fine trimmer value for pin4 of BYTE5

18.11.2.405 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE5_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE5: This should be programmed to RX fine trimmer value for pin8 of BYTE5

18.11.2.406 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE6_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin3 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa60 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE6: This should be programmed to RX fine trimmer value for pin3 of BYTE6
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE6: This should be programmed to RX fine trimmer value for pin2 of BYTE6

Bit	Reset	Description
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE6: This should be programmed to RX fine trimmer value for pin1 of BYTE6
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE6: This should be programmed to RX fine trimmer value for pin0 of BYTE6

18.11.2.407 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE6_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa64 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE6: This should be programmed to RX fine trimmer value for pin7 of BYTE6
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE6: This should be programmed to RX fine trimmer value for pin6 of BYTE6
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE6: This should be programmed to RX fine trimmer value for pin5 of BYTE6
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE6: This should be programmed to RX fine trimmer value for pin4 of BYTE6

18.11.2.408 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE6_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE6: This should be programmed to RX fine trimmer value for pin8 of BYTE6

18.11.2.409 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE7_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin3 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa70 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_BYTE7: This should be programmed to RX fine trimmer value for pin3 of BYTE7
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_BYTE7: This should be programmed to RX fine trimmer value for pin2 of BYTE7
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_BYTE7: This should be programmed to RX fine trimmer value for pin1 of BYTE7
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_BYTE7: This should be programmed to RX fine trimmer value for pin0 of BYTE7

18.11.2.410 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE7_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa74 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_BYTE7: This should be programmed to RX fine trimmer value for pin7 of BYTE7
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_BYTE7: This should be programmed to RX fine trimmer value for pin6 of BYTE7
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_BYTE7: This should be programmed to RX fine trimmer value for pin5 of BYTE7
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_BYTE7: This should be programmed to RX fine trimmer value for pin4 of BYTE7

18.11.2.411 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_BYTE7_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa78 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_BYTE7: This should be programmed to RX fine trimmer value for pin8 of BYTE7

18.11.2.412 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD0_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa80 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_CMD0: This should be programmed to TX fine trimmer value for pin3 of CMD0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_CMD0: This should be programmed to TX fine trimmer value for pin2 of CMD0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_CMD0: This should be programmed to TX fine trimmer value for pin1 of CMD0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_CMD0: This should be programmed to TX fine trimmer value for pin0 of CMD0

18.11.2.413 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD0_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa84 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_CMD0: This should be programmed to TX fine trimmer value for pin7 of CMD0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_CMD0: This should be programmed to TX fine trimmer value for pin6 of CMD0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_CMD0: This should be programmed to TX fine trimmer value for pin5 of CMD0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_CMD0: This should be programmed to TX fine trimmer value for pin4 of CMD0

18.11.2.414 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD0_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_CMD0: This should be programmed to TX fine trimmer value for pin8 of CMD0

18.11.2.415 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD1_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa90 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_CMD1: This should be programmed to TX fine trimmer value for pin3 of CMD1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_CMD1: This should be programmed to TX fine trimmer value for pin2 of CMD1
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_CMD1: This should be programmed to TX fine trimmer value for pin1 of CMD1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_CMD1: This should be programmed to TX fine trimmer value for pin0 of CMD1

18.11.2.416 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD1_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa94 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_CMD1: This should be programmed to TX fine trimmer value for pin7 of CMD1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_CMD1: This should be programmed to TX fine trimmer value for pin6 of CMD1
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_CMD1: This should be programmed to TX fine trimmer value for pin5 of CMD1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_CMD1: This should be programmed to TX fine trimmer value for pin4 of CMD1

18.11.2.417 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD1_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xa98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_CMD1: This should be programmed to TX fine trimmer value for pin8 of CMD1

18.11.2.418 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD2_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xaa0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_CMD2: This should be programmed to TX fine trimmer value for pin3 of CMD2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_CMD2: This should be programmed to TX fine trimmer value for pin2 of CMD2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_CMD2: This should be programmed to TX fine trimmer value for pin1 of CMD2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_CMD2: This should be programmed to TX fine trimmer value for pin0 of CMD2

18.11.2.419 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD2_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xaa4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_CMD2: This should be programmed to TX fine trimmer value for pin7 of CMD2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_CMD2: This should be programmed to TX fine trimmer value for pin6 of CMD2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_CMD2: This should be programmed to TX fine trimmer value for pin5 of CMD2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_CMD2: This should be programmed to TX fine trimmer value for pin4 of CMD2

18.11.2.420 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD2_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xaa8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_CMD2: This should be programmed to TX fine trimmer value for pin8 of CMD2

18.11.2.421 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD3_0_0

Rank0 IB SHORT DQ DDLL value for pin0 to pin1 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xab0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN3_CMD3: This should be programmed to TX fine trimmer value for pin3 of CMD3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN2_CMD3: This should be programmed to TX fine trimmer value for pin2 of CMD3

Bit	Reset	Description
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN1_CMD3: This should be programmed to TX fine trimmer value for pin1 of CMD3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN0_CMD3: This should be programmed to TX fine trimmer value for pin0 of CMD3

18.11.2.422 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD3_1_0

Rank0 IB SHORT DQ DDLL value for pin4 to pin7 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xab4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN7_CMD3: This should be programmed to TX fine trimmer value for pin7 of CMD3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN6_CMD3: This should be programmed to TX fine trimmer value for pin6 of CMD3
14:8	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN5_CMD3: This should be programmed to TX fine trimmer value for pin5 of CMD3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN4_CMD3: This should be programmed to TX fine trimmer value for pin4 of CMD3

18.11.2.423 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK0_CMD3_2_0

Rank0 IB SHORT DQ DDLL value for pin8 to pin10 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xab8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK0_PIN8_CMD3: This should be programmed to TX fine trimmer value for pin8 of CMD3

18.11.2.424 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE0_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb00 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE0: This should be programmed to RX fine trimmer value for pin3 of BYTE0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE0: This should be programmed to RX fine trimmer value for pin2 of BYTE0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE0: This should be programmed to RX fine trimmer value for pin1 of BYTE0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE0: This should be programmed to RX fine trimmer value for pin0 of BYTE0

18.11.2.425 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE0_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb04 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE0: This should be programmed to RX fine trimmer value for pin7 of BYTE0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE0: This should be programmed to RX fine trimmer value for pin6 of BYTE0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE0: This should be programmed to RX fine trimmer value for pin5 of BYTE0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE0: This should be programmed to RX fine trimmer value for pin4 of BYTE0

18.11.2.426 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE0_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE0: This should be programmed to RX fine trimmer value for pin8 of BYTE0

18.11.2.427 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE1_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb10 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE1: This should be programmed to RX fine trimmer value for pin3 of BYTE1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE1: This should be programmed to RX fine trimmer value for pin2 of BYTE1
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE1: This should be programmed to RX fine trimmer value for pin1 of BYTE1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE1: This should be programmed to RX fine trimmer value for pin0 of BYTE1

18.11.2.428 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE1_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb14 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE1: This should be programmed to RX fine trimmer value for pin7 of BYTE1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE1: This should be programmed to RX fine trimmer value for pin6 of BYTE1
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE1: This should be programmed to RX fine trimmer value for pin5 of BYTE1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE1: This should be programmed to RX fine trimmer value for pin4 of BYTE1

18.11.2.429 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE1_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0xb18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE1: This should be programmed to RX fine trimmer value for pin8 of BYTE1

18.11.2.430 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE2_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0xb20 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE2: This should be programmed to RX fine trimmer value for pin3 of BYTE2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE2: This should be programmed to RX fine trimmer value for pin2 of BYTE2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE2: This should be programmed to RX fine trimmer value for pin1 of BYTE2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE2: This should be programmed to RX fine trimmer value for pin0 of BYTE2

18.11.2.431 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE2_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb24 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE2: This should be programmed to RX fine trimmer value for pin7 of BYTE2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE2: This should be programmed to RX fine trimmer value for pin6 of BYTE2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE2: This should be programmed to RX fine trimmer value for pin5 of BYTE2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE2: This should be programmed to RX fine trimmer value for pin4 of BYTE2

18.11.2.432 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE2_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE2: This should be programmed to RX fine trimmer value for pin8 of BYTE2

18.11.2.433 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE3_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb30 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE3: This should be programmed to RX fine trimmer value for pin3 of BYTE3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE3: This should be programmed to RX fine trimmer value for pin2 of BYTE3
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE3: This should be programmed to RX fine trimmer value for pin1 of BYTE3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE3: This should be programmed to RX fine trimmer value for pin0 of BYTE3

18.11.2.434 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE3_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb34 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE3: This should be programmed to RX fine trimmer value for pin7 of BYTE3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE3: This should be programmed to RX fine trimmer value for pin6 of BYTE3
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE3: This should be programmed to RX fine trimmer value for pin5 of BYTE3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE3: This should be programmed to RX fine trimmer value for pin4 of BYTE3

18.11.2.435 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE3_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE3: This should be programmed to RX fine trimmer value for pin8 of BYTE3

18.11.2.436 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE4_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb40 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE4: This should be programmed to RX fine trimmer value for pin3 of BYTE4
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE4: This should be programmed to RX fine trimmer value for pin2 of BYTE4

Bit	Reset	Description
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE4: This should be programmed to RX fine trimmer value for pin1 of BYTE4
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE4: This should be programmed to RX fine trimmer value for pin0 of BYTE4

18.11.2.437 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE4_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb44 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE4: This should be programmed to RX fine trimmer value for pin7 of BYTE4
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE4: This should be programmed to RX fine trimmer value for pin6 of BYTE4
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE4: This should be programmed to RX fine trimmer value for pin5 of BYTE4
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE4: This should be programmed to RX fine trimmer value for pin4 of BYTE4

18.11.2.438 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE4_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte4 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE4: This should be programmed to RX fine trimmer value for pin8 of BYTE4

18.11.2.439 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE5_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb50 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE5: This should be programmed to RX fine trimmer value for pin3 of BYTE5
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE5: This should be programmed to RX fine trimmer value for pin2 of BYTE5
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE5: This should be programmed to RX fine trimmer value for pin1 of BYTE5
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE5: This should be programmed to RX fine trimmer value for pin0 of BYTE5

18.11.2.440 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE5_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb54 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE5: This should be programmed to RX fine trimmer value for pin7 of BYTE5
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE5: This should be programmed to RX fine trimmer value for pin6 of BYTE5
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE5: This should be programmed to RX fine trimmer value for pin5 of BYTE5
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE5: This should be programmed to RX fine trimmer value for pin4 of BYTE5

18.11.2.441 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE5_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte5 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE5: This should be programmed to RX fine trimmer value for pin8 of BYTE5

18.11.2.442 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE6_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0xb60 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE6: This should be programmed to RX fine trimmer value for pin3 of BYTE6
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE6: This should be programmed to RX fine trimmer value for pin2 of BYTE6
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE6: This should be programmed to RX fine trimmer value for pin1 of BYTE6
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE6: This should be programmed to RX fine trimmer value for pin0 of BYTE6

18.11.2.443 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE6_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb64 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE6: This should be programmed to RX fine trimmer value for pin7 of BYTE6
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE6: This should be programmed to RX fine trimmer value for pin6 of BYTE6
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE6: This should be programmed to RX fine trimmer value for pin5 of BYTE6
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE6: This should be programmed to RX fine trimmer value for pin4 of BYTE6

18.11.2.444 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE6_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte6 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE6: This should be programmed to RX fine trimmer value for pin8 of BYTE6

18.11.2.445 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE7_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin3 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb70 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_BYTE7: This should be programmed to RX fine trimmer value for pin3 of BYTE7
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_BYTE7: This should be programmed to RX fine trimmer value for pin2 of BYTE7
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_BYTE7: This should be programmed to RX fine trimmer value for pin1 of BYTE7
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_BYTE7: This should be programmed to RX fine trimmer value for pin0 of BYTE7

18.11.2.446 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE7_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb74 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_BYTE7: This should be programmed to RX fine trimmer value for pin7 of BYTE7
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_BYTE7: This should be programmed to RX fine trimmer value for pin6 of BYTE7
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_BYTE7: This should be programmed to RX fine trimmer value for pin5 of BYTE7
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_BYTE7: This should be programmed to RX fine trimmer value for pin4 of BYTE7

18.11.2.447 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_BYTE7_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of byte7 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb78 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_BYTE7: This should be programmed to RX fine trimmer value for pin8 of BYTE7

18.11.2.448 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD0_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin1 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers..

Offset: 0xb80 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_CMD0: This should be programmed to TX fine trimmer value for pin3 of CMD0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_CMD0: This should be programmed to TX fine trimmer value for pin2 of CMD0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_CMD0: This should be programmed to TX fine trimmer value for pin1 of CMD0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_CMD0: This should be programmed to TX fine trimmer value for pin0 of CMD0

18.11.2.449 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD0_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb84 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_CMD0: This should be programmed to TX fine trimmer value for pin7 of CMD0
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_CMD0: This should be programmed to TX fine trimmer value for pin6 of CMD0
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_CMD0: This should be programmed to TX fine trimmer value for pin5 of CMD0
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_CMD0: This should be programmed to TX fine trimmer value for pin4 of CMD0

18.11.2.450 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD0_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of cmd0 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_CMD0: This should be programmed to TX fine trimmer value for pin8 of CMD0

18.11.2.451 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD1_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin1 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb90 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_CMD1: This should be programmed to TX fine trimmer value for pin3 of CMD1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_CMD1: This should be programmed to TX fine trimmer value for pin2 of CMD1

Bit	Reset	Description
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_CMD1: This should be programmed to TX fine trimmer value for pin1 of CMD1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_CMD1: This should be programmed to TX fine trimmer value for pin0 of CMD1

18.11.2.452 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD1_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb94 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_CMD1: This should be programmed to TX fine trimmer value for pin7 of CMD1
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_CMD1: This should be programmed to TX fine trimmer value for pin6 of CMD1
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_CMD1: This should be programmed to TX fine trimmer value for pin5 of CMD1
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_CMD1: This should be programmed to TX fine trimmer value for pin4 of CMD1

18.11.2.453 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD1_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of cmd1 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xb98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_CMD1: This should be programmed to TX fine trimmer value for pin8 of CMD1

18.11.2.454 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD2_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin1 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xba0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_CMD2: This should be programmed to TX fine trimmer value for pin3 of CMD2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_CMD2: This should be programmed to TX fine trimmer value for pin2 of CMD2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_CMD2: This should be programmed to TX fine trimmer value for pin1 of CMD2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_CMD2: This should be programmed to TX fine trimmer value for pin0 of CMD2

18.11.2.455 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD2_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xba4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_CMD2: This should be programmed to TX fine trimmer value for pin7 of CMD2
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_CMD2: This should be programmed to TX fine trimmer value for pin6 of CMD2
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_CMD2: This should be programmed to TX fine trimmer value for pin5 of CMD2
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_CMD2: This should be programmed to TX fine trimmer value for pin4 of CMD2

18.11.2.456 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD2_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of cmd2 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xba8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_CMD2: This should be programmed to TX fine trimmer value for pin8 of CMD2

18.11.2.457 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD3_0_0

Rank1 IB SHORT DQ DDLL value for pin0 to pin1 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xbb0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN3_CMD3: This should be programmed to TX fine trimmer value for pin3 of CMD3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN2_CMD3: This should be programmed to TX fine trimmer value for pin2 of CMD3
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN1_CMD3: This should be programmed to TX fine trimmer value for pin1 of CMD3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN0_CMD3: This should be programmed to TX fine trimmer value for pin0 of CMD3

18.11.2.458 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD3_1_0

Rank1 IB SHORT DQ DDLL value for pin4 to pin7 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xbb4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN7_CMD3: This should be programmed to TX fine trimmer value for pin7 of CMD3
22:16	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN6_CMD3: This should be programmed to TX fine trimmer value for pin6 of CMD3
14:8	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN5_CMD3: This should be programmed to TX fine trimmer value for pin5 of CMD3
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN4_CMD3: This should be programmed to TX fine trimmer value for pin4 of CMD3

18.11.2.459 EMC_PMACRO_IB_DDLL_SHORT_DQ_RANK1_CMD3_2_0

Rank1 IB SHORT DQ DDLL value for pin8 to pin10 of cmd3 brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Offset: 0xbb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	IB_DDLL_SHORT_DQ_RANK1_PIN8_CMD3: This should be programmed to TX fine trimmer value for pin8 of CMD3

18.11.2.460 EMC_PMACRO_IB_VREF_DQ_0_0

IB DQ Receiver Vref for Byte0 to Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xbe0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_VREF_DQ_BYTE3: [PMC] This should be programmed to IB vref value for DQ of BYTE3
22:16	0x0	IB_VREF_DQ_BYTE2: [PMC] This should be programmed to IB vref value for DQ of BYTE2
14:8	0x0	IB_VREF_DQ_BYTE1: [PMC] This should be programmed to IB vref value for DQ of BYTE1
6:0	0x0	IB_VREF_DQ_BYTE0: [PMC] This should be programmed to IB vref value for DQ of BYTE0

18.11.2.461 EMC_PMACRO_IB_VREF_DQ_1_0

IB DQ Receiver Vref for Byte4 to Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xbe4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_VREF_DQ_BYTE7: [PMC] This should be programmed to IB vref value for DQ of BYTE7
22:16	0x0	IB_VREF_DQ_BYTE6: [PMC] This should be programmed to IB vref value for DQ of BYTE6
14:8	0x0	IB_VREF_DQ_BYTE5: [PMC] This should be programmed to IB vref value for DQ of BYTE5
6:0	0x0	IB_VREF_DQ_BYTE4: [PMC] This should be programmed to IB vref value for DQ of BYTE4

18.11.2.462 EMC_PMACRO_IB_VREF_DQ_2_0

IB DQ Receiver Vref for cmd0 to cmd3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xbe8 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_VREF_DQ_CMD3: This should be programmed to IB vref value for DQ of CMD3
22:16	0x0	IB_VREF_DQ_CMD2: This should be programmed to IB vref value for DQ of CMD2
14:8	0x0	IB_VREF_DQ_CMD1: This should be programmed to IB vref value for DQ of CMD1
6:0	0x0	IB_VREF_DQ_CMD0: This should be programmed to IB vref value for DQ of CMD0

18.11.2.463 EMC_PMACRO_IB_VREF_DQS_0_0

IB DQS Receiver Vref for Byte0 to Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xbf0 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_VREF_DQS_BYTE3: [PMC] This should be programmed to IB vref value for DQS of BYTE3
22:16	0x0	IB_VREF_DQS_BYTE2: [PMC] This should be programmed to IB vref value for DQS of BYTE2
14:8	0x0	IB_VREF_DQS_BYTE1: [PMC] This should be programmed to IB vref value for DQS of BYTE1
6:0	0x0	IB_VREF_DQS_BYTE0: [PMC] This should be programmed to IB vref value for DQS of BYTE0

18.11.2.464 EMC_PMACRO_IB_VREF_DQS_1_0

IB DQS Receiver Vref for Byte4 to Byte7

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xbf4 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_VREF_DQS_BYTE7: [PMC] This should be programmed to IB vref value for DQS of BYTE7
22:16	0x0	IB_VREF_DQS_BYTE6: [PMC] This should be programmed to IB vref value for DQS of BYTE6
14:8	0x0	IB_VREF_DQS_BYTE5: [PMC] This should be programmed to IB vref value for DQS of BYTE5
6:0	0x0	IB_VREF_DQS_BYTE4: [PMC] This should be programmed to IB vref value for DQS of BYTE4

18.11.2.465 EMC_PMACRO_IB_VREF_DQS_2_0

IB DQS Receiver Vref for cmd0 to cmd3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xbf8 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	IB_VREF_DQS_CMD3: This should be programmed to IB vref value for DQS of CMD3
22:16	0x0	IB_VREF_DQS_CMD2: This should be programmed to IB vref value for DQS of CMD2
14:8	0x0	IB_VREF_DQS_CMD1: This should be programmed to IB vref value for DQS of CMD1
6:0	0x0	IB_VREF_DQS_CMD0: This should be programmed to IB vref value for DQS of CMD0

18.11.2.466 EMC_PMACRO_DDLL_LONG_CMD_0_0

DLL long trimmer value for cke0 and cke1 pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	DDLL_LONG_CMD_CKE1: [PMC] This should be programmed to CKE trimmer value for CKE1 brick
10:0	0x0	DDLL_LONG_CMD_CKE0: [PMC] This should be programmed to CKE trimmer value for CKE0 brick

18.11.2.467 EMC_PMACRO_DDLL_LONG_CMD_1_0

DLL long trimmer value for cke2 and cke3 pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc04 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	DDLL_LONG_CMD_CKE3: [PMC] This should be programmed to CKE trimmer value for CKE3 brick
10:0	0x0	DDLL_LONG_CMD_CKE2: [PMC] This should be programmed to CKE trimmer value for CKE2 brick

18.11.2.468 EMC_PMACRO_DDLL_LONG_CMD_2_0

DLL long trimmer value for cke4 and cke5 pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	DDLL_LONG_CMD_CKE5: [PMC] This should be programmed to CKE trimmer value for CKE5 brick
10:0	0x0	DDLL_LONG_CMD_CKE4: [PMC] This should be programmed to CKE trimmer value for CKE4 brick

18.11.2.469 EMC_PMACRO_DDLL_LONG_CMD_3_0

DLL long trimmer value for cke6 and cke7 pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc0c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	DDLL_LONG_CMD_CKE7: [PMC] This should be programmed to CKE trimmer value for CKE7 brick
10:0	0x0	DDLL_LONG_CMD_CKE6: [PMC] This should be programmed to CKE trimmer value for CKE6 brick

18.11.2.470 EMC_PMACRO_DDLL_LONG_CMD_4_0

DLL long trimmer value for Reset pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	DDLL_LONG_CMD_MISC0: This should be programmed to RESET trimmer value for MISC brick. Not intended to be used
10:0	0x0	DDLL_LONG_CMD_RESET: [PMC] This should be programmed to RESET trimmer value for RESET brick

18.11.2.471 EMC_PMACRO_DDLL_LONG_CMD_5_0

DLL long trimmer value for misc pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	DDLL_LONG_CMD_MISC2: This should be programmed to RESET trimmer value for MISC brick. Not intended to be used
10:0	0x0	DDLL_LONG_CMD_MISC1: This should be programmed to RESET trimmer value for MISC brick. Not intended to be used

18.11.2.472 EMC_PMACRO_DDLL_SHORT_CMD_0_0

DLL short trimmer value for cke0 to cke3 pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc20 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	**DDLL_SHORT_CMD_CKE3: [PMC] This should be programmed to CKE fine trimmer value for CKE3 brick**
22:16	0x0	DDLL_SHORT_CMD_CKE2: [PMC] This should be programmed to CKE fine trimmer value for CKE2 brick
14:8	0x0	DDLL_SHORT_CMD_CKE1: [PMC] This should be programmed to CKE fine trimmer value for CKE1 brick
6:0	0x0	DDLL_SHORT_CMD_CKE0: [PMC] This should be programmed to CKE fine trimmer value for CKE0 brick

18.11.2.473 EMC_PMACRO_DDLL_SHORT_CMD_1_0

DLL short trimmer value for cke4 to cke7 pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc24 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	DDLL_SHORT_CMD_CKE7: [PMC] This should be programmed to CKE fine trimmer value for CKE7 brick
22:16	0x0	DDLL_SHORT_CMD_CKE6: [PMC] This should be programmed to CKE fine trimmer value for CKE6 brick
14:8	0x0	DDLL_SHORT_CMD_CKE5: [PMC] This should be programmed to CKE fine trimmer value for CKE5 brick
6:0	0x0	DDLL_SHORT_CMD_CKE4: [PMC] This should be programmed to CKE fine trimmer value for CKE4 brick

18.11.2.474 EMC_PMACRO_DDLL_SHORT_CMD_2_0

DLL short trimmer value for Reset pin

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc28 | Read/Write: R/W | Reset: 0x00000000 (0bx0000000x0000000x0000000x0000000)

Bit	Reset	Description
30:24	0x0	DDLL_SHORT_CMD_MISC2: This should be programmed to MISC fine trimmer value for MISC2 brick. Not intended to be used
22:16	0x0	DDLL_SHORT_CMD_MISC1: This should be programmed to MISC fine trimmer value for MISC1 brick. Not intended to be used
14:8	0x0	DDLL_SHORT_CMD_MISC0: This should be programmed to MISC fine trimmer value for MISC0 brick. Not intended to be used
6:0	0x0	DDLL_SHORT_CMD_RESET: [PMC] This should be programmed to RESET fine trimmer value for Reset pin

18.11.2.475 EMC_PMACRO_CFG_PM_GLOBAL_0_0

Global reg for pad macro control

Offset: 0xc30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx000000000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
27	0x0	DISABLE_CFG_CMD3: Disables configuration reads/writes to CMD3 pad macro, except to this register
26	0x0	DISABLE_CFG_CMD2: Disables configuration reads/writes to CMD2 pad macro, except to this register
25	0x0	DISABLE_CFG_CMD1: Disables configuration reads/writes to CMD1 pad macro, except to this register
24	0x0	DISABLE_CFG_CMD0: Disables configuration reads/writes to CMD0 pad macro, except to this register
23	0x0	DISABLE_CFG_BYTE7: Disables configuration reads/writes to BYTE7 pad macro, except to this register
22	0x0	DISABLE_CFG_BYTE6: Disables configuration reads/writes to BYTE6 pad macro, except to this register
21	0x0	DISABLE_CFG_BYTE5: Disables configuration reads/writes to BYTE5 pad macro, except to this register
20	0x0	DISABLE_CFG_BYTE4: Disables configuration reads/writes to BYTE4 pad macro, except to this register
19	0x0	DISABLE_CFG_BYTE3: Disables configuration reads/writes to BYTE3 pad macro, except to this register
18	0x0	DISABLE_CFG_BYTE2: Disables configuration reads/writes to BYTE2 pad macro, except to this register
17	0x0	DISABLE_CFG_BYTE1: Disables configuration reads/writes to BYTE1 pad macro, except to this register
16	0x0	DISABLE_CFG_BYTE0: Disables configuration reads/writes to BYTE0 pad macro, except to this register

18.11.2.476 EMC_PMACRO_VTTGEN_CTRL_0_0

VTTGEN control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc34 | Read/Write: R/W | Reset: 0x00030202 (0bxxxxxxxxxxxx0011xxxx0010xxxx0010)

Bit	Reset	Description
19:16	0x3	VTT_VDDA_LVL: [PMC] Level select for VDDA
11:8	0x2	VTT_VAUXP_LVL: [PMC] Level select for VAUXP
3:0	0x2	VTT_VCLAMP_LVL: [PMC] Level select for VCLAMP

18.11.2.477 EMC_PMACRO_VTTGEN_CTRL_1_0

VTTGEN control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:18	0x0	VTT_SPARE: [PMC] Reserved
17:16	0x0	VTT_VDDA_WB_CTRL: [PMC] Reserved
15:14	0x0	VTT_VAUXP_WB_CTRL:[PMC] Reserved
13:12	0x0	VTT_VCLAMP_WB_CTRL: [PMC] Reserved
11:8	0x0	VTT_VDDA_CTRL: [PMC] Reserved. Fast droop recovery or stability

Bit	Reset	Description
7:4	0x0	VTT_VAUXP_CTRL: [PMC] Reserved. Fast droop recovery or stability
3:0	0x0	VTT_VCLAMP_CTRL: [PMC] Reserved Fast droop recovery or stability

18.11.2.478 EMC_PMACRO_BG_BIAS_CTRL_0_0

BG Bias control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc3c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011x100)

Bit	Reset	Description
6:4	0x3	BG_SETUP: [PMC] Temperature coefficient control
2	0x1	BGLP_E_PWRD: [PMC]
1	0x0	BG_MODE: [PMC] Set regulator mode, 0: High performance mode, 1: Lower power mode
0	0x0	BG_E_PWRD: [PMC] power down regulator

18.11.2.479 EMC_PMACRO_PAD_CFG_CTRL_0

Config control register for pads

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc40 | Read/Write: R/W | Reset: 0x00000000 (0bx000xx00xx00xx00xx0x000xx00xxx00)

Bit	Reset	Description
30:28	0x0	PAD_MACRO_QUSE_MODE: [PMC] 0 = DIRECT_QUSE 1 = RESERVED 2 = REGULAR_WO_DIV2 3 = REGULAR 4 = RESERVED1
25	0x0	TX_E_DDLL_TCLK: [PMC]
24	0x0	E_TX_NMOS_DRVUP: [PMC]
21:20	0x0	TX_DCC_MODE: [PMC]
17:16	0x0	MEM_MODE: [PMC] Type of DDR memory used for misc. internal controls, 00 = LPDDR2/3, 01 = SDDR3/SDDR3L, 10 = LPDDR4, 11= Do not use.
13	0x0	TX_SEL_MV_CYCLE: [PMC] Select timing closure cycle for Multi-Voltage timing path. 0 = 1 cycle, 1 = 2 cycle
11:10	0x0	CMD_TX_EBOOST_PU_MODE: [PMC] Select DQ pull up pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting,11 = 4X edge boosting
9	0x0	E_PWRD: [PMC] Active High to disable DC current paths within entire brick cell
6:5	0x0	CMD_TX_EBOOST_PD_MODE: [PMC] Select DQ pull down pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting,11 = 4X edge boosting
1:0	0x0	QUSE_MODE: [PMC] quse mode for pad

18.11.2.480 EMC_PMACRO_ZCTRL_0

Driver/Receiver ZCTRL settings for all IOBRICKs

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27:26	0x0	DATA_DQS_TX_DRVDN_ZCTRL: [PMC] Drive pull-down impedance calibration select for DQS. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
25:24	0x0	DATA_DQ_TX_DRVDN_ZCTRL: [PMC] Drive pull-down impedance calibration select for DQ. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
23:22	0x0	DATA_DQS_TX_DRVUP_ZCTRL: [PMC] Drive pull-up impedance calibration select for DQS. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
21:20	0x0	DATA_DQ_TX_DRVUP_ZCTRL: [PMC] Drive pull-up impedance calibration select for DQ. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
19:18	0x0	CMD_DQS_TX_DRVDN_ZCTRL: [PMC] Drive pull-down impedance calibration select for CK. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
17:16	0x0	CMD_DQ_TX_DRVDN_ZCTRL: [PMC] Drive pull-down impedance calibration select for CA. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
15:14	0x0	CMD_DQS_TX_DRVUP_ZCTRL: [PMC] Drive pull-up impedance calibration select for CK. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
13:12	0x0	CMD_DQ_TX_DRVUP_ZCTRL: [PMC] Drive pull-up impedance calibration select for CA. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
11:10	0x0	DQS_RX_DRVDN_TERM_ZCTRL: [PMC] Termination pull-down impedance calibration select for DQS. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
9:8	0x0	DQ_RX_DRVDN_TERM_ZCTRL: [PMC] Termination pull-down impedance calibration select for DQ. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
7:6	0x0	DQS_RX_DRVUP_TERM_ZCTRL: [PMC] Termination pull-up impedance calibration select for DQS. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
5:4	0x0	DQ_RX_DRVUP_TERM_ZCTRL: [PMC] Termination pull-up impedance calibration select for DQ. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
3:2	0x0	CMD_TX_DRVDN_ZCTRL: [PMC] Drive pull-down impedance calibration select for CMD pad of DQ/CA brick. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms
1:0	0x0	CMD_TX_DRVUP_ZCTRL: [PMC] Drive pull-up impedance calibration select for CMD pad of DQ/CA brick. 00 = 40ohms, 01 = 60ohms, 10 = 80ohms, 11 = 120 ohms

18.11.2.481 EMC_PMACRO_RX_TERM_0

Receive termination strength control register for DQ brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc48 | Read/Write: R/W | Reset: 0x1f1f1f1f (0bxx011111xx011111xx011111xx011111)

Bit	Reset	Description
29:24	0x1f	DQS_RX_DRVDN_TERM: [PMC] RX pull-down termination strength code for DQS.
21:16	0x1f	DQS_RX_DRVUP_TERM: [PMC] RX pull-up termination strength code for DQS.
13:8	0x1f	DQ_RX_DRVDN_TERM: [PMC] RX pull-down termination strength code for DQ.

Bit	Reset	Description
5:0	0x1f	DQ_RX_DRVUP_TERM: [PMC] RX pull-up termination strength code for DQ.

18.11.2.482 EMC_PMACRO_CMD_TX_DRV_0

Transmit drive strength control reg for CMD pad in DQ/CA brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc4c | Read/Write: R/W | Reset: 0x00001f1f (0bxxxxxxxxxxxxxxxxxxxx011111xx011111)

Bit	Reset	Description
13:8	0x1f	CMD_TX_DRVDN: [PMC] TX pull-down drive strength code for CMD.
5:0	0x1f	CMD_TX_DRVUP: [PMC] TX pull-up drive strength code for CMD.

18.11.2.483 EMC_PMACRO_CMD_PAD_RX_CTRL_0

Receiver mode control for cmd pad

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc50 | Read/Write: R/W | Reset: 0x00008000 (0bxxx00000x00000001000x000xx00x000)

Bit	Reset	Description
28:24	0x0	CMD_DQ_RX_CTRL: [PMC] Active high for differential amplifier to adjust bias current
22	0x0	CMD_RX_E_DIRECT_ZI: [PMC]
21	0x0	CMD_RX_E_IBIAS_PWRD: [PMC]
20:16	0x0	CMD_DQS_RX_CTRL: [PMC] Active high for differential amplifier to adjust bias current
15	0x1	CMD_EINPUT_FORCE_ON: [PMC] Active high for enabling the static einput -- enable at reset to clear QUSE serializer flops in BRICK
14	0x0	CMD_DQ_RX_E_RDIV: [PMC] Active high for differential amplifier to enable resistor divider
13	0x0	CMD_DQ_RX_E_DAMP_DFE: [PMC] Active high to enable RX differential amplify DFE Circuit
12	0x0	CMD_DQS_RX_E_DIFF_MODE: [PMC] Enable strobe RX in differential mode; 10ns latency for mode switch. 1 = IOP-ION, 0 = IOP-VREF
10:8	0x0	CMD_DQ_RX_DAMP_DFE_CTRL: [PMC] Selection for different DFE modes.
5:4	0x0	CMD_DQS_RX_MODE: [PMC] Set front-end RX type. 00: Schmitt common mode @ 50% of VDDP, 10: DAMP common mode @ 50% of VDDP, 11: DAMP (Differential Amplifier) common mode @ 16.67% 0 = DQS_SCHMITT 1 = RESERVED 2 = DQS_DIFF_MODE_50 3 = DQS_DIFF_MODE_15

2:0	0x0	CMD_DQ_RX_MODE: [PMC] Set front-end RX type. 0: Schmitt common mode @ 50% of VDDP, 2: DAMP common mode @ 50% of VDDP, 3: DAMP (Differential Amplifier) common mode @ 16.67%, 4: HSSA common mode @ 50% of VDDP, 5: HSSA common mode @ 16.67% of VDDP, 7: LSSA @<50%. others are RFU 0 = DQ_SCHMITT 1 = RESERVED 2 = DQ_DIFF_MODE_50 3 = DQ_DIFF_MODE_15 4 = DQ_HSSA_MODE_50 5 = DQ_HSSA_MODE_15 6 = RESERVED1 7 = DQ_LSSA_MODE

18.11.2.484 EMC_PMACRO_DATA_PAD_RX_CTRL_0

Receiver mode control for data pad

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc54 | Read/Write: R/W | Reset: 0x00008000 (0bxxx00000x00000001000x000xx00x000)

Bit	Reset	Description
28:24	0x0	DATA_DQ_RX_CTRL: [PMC] Active high for differential amplifier to adjust bias current
22	0x0	DATA_RX_E_DIRECT_ZI: [PMC]
21	0x0	DATA_RX_E_IBIAS_PWRD: [PMC]
20:16	0x0	DATA_DQS_RX_CTRL: [PMC] Active high for differential amplifier to adjust bias current
15	0x1	DATA_EINPUT_FORCE_ON: [PMC] Active high for enabling the static einput -- enable at reset to clear QUSE serializer flops in BRICK
14	0x0	DATA_DQ_RX_E_RDIV: [PMC] Active high for differential amplifier to enable resistor divider
13	0x0	DATA_DQ_RX_E_DAMP_DFE: [PMC] Active high to enable RX differential amplify DFE Circuit
12	0x0	DATA_DQS_RX_E_DIFF_MODE: [PMC] Enable strobe RX in differential mode; 10ns latency for mode switch. 1 = IOP-ION, 0 = IOP-VREF
10:8	0x0	DATA_DQ_RX_DAMP_DFE_CTRL: [PMC] Selection for different DFE modes.
5:4	0x0	DATA_DQS_RX_MODE: [PMC] Set front-end RX type. 00: Schmitt common mode @ 50% of VDDP, 10: DAMP common mode @ 50% of VDDP, 11: DAMP (Differential Amplifier) common mode @ 16.67% 0 = DQS_SCHMITT 1 = DQS_DIFF_MODE_50 2 = DQS_DIFF_MODE_15 3 = RESERVED
2:0	0x0	DATA_DQ_RX_MODE: [PMC] Set front-end RX type. 0: Schmitt common mode @ 50% of VDDP, 2: DAMP common mode @ 50% of VDDP, 3: DAMP (Differential Amplifier) common mode @ 16.67%, 4: HSSA common mode @ 50% of VDDP, 5: HSSA common mode @ 16.67% of VDDP, 7: LSSA @<50%. others are RFU 0 = DQ_SCHMITT 1 = RESERVED 2 = DQ_DIFF_MODE_50 3 = DQ_DIFF_MODE_15 4 = DQ_HSSA_MODE_50 5 = DQ_HSSA_MODE_15 6 = RESERVED1 7 = DQ_LSSA_MODE

18.11.2.485 EMC_PMACRO_CMD_RX_TERM_MODE_0

Receiver termination mode control for cmd pad

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc58 | Read/Write: R/W | Reset: 0x00002000 (0bxxxxxxxxxxxxxxxxxx10xx00xx00xx00)

Bit	Reset	Description
13	0x1	CMD_E_PWRD_RX: [PMC] Active High to disable RX DC current paths within entire brick cell (for outbound only IOBRICKs).
12	0x0	CMD_DDLL_QU_SEL_M2CLK: [PMC] Select clock for QUSE DDLL. 0 = QUSE_IN for DQBRICK, 1 = M2CLKP/N for ADBRICK
9:8	0x0	CMD_DQSN_RX_TERM_MODE: [PMC] 10 = Enable pull up termination for RX, 11= Enable Center Tap Termination for RX
5:4	0x0	CMD_DQSP_RX_TERM_MODE: [PMC] 00 = HIZ, 01 = Enable pull down termination for RX
1:0	0x0	CMD_DQ_RX_TERM_MODE: [PMC] Enable Driver as receiver ODT or CTT termination.

18.11.2.486 EMC_PMACRO_DATA_RX_TERM_MODE_0

Receiver termination mode control for data pad

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc5c | Read/Write: R/W | Reset: 0x00002000 (0bxxxxxxxxxxxxxxxxxx10xx00xx00xx00)

Bit	Reset	Description
13	0x1	DATA_E_PWRD_RX: [PMC] Active High to disable RX DC current paths within entire brick cell (for outbound only IOBRICKs).
12	0x0	DATA_DDLL_QU_SEL_M2CLK: [PMC] Select clock for QUSE DDLL. 0 = QUSE_IN for DQBRICK, 1 = M2CLKP/N for ADBRICK
9:8	0x0	DATA_DQSN_RX_TERM_MODE: [PMC] 10 = Enable pull up termination for RX, 11= Enable Center Tap Termination for RX
5:4	0x0	DATA_DQSP_RX_TERM_MODE: [PMC] 00 = HIZ, 01 = Enable pull down termination for RX
1:0	0x0	DATA_DQ_RX_TERM_MODE: [PMC] Enable Driver as receiver ODT or CTT termination.

18.11.2.487 EMC_PMACRO_CMD_PAD_TX_CTRL_0

OB control for cmd pad

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc60 | Read/Write: R/W | Reset: 0x02000000 (0bxxxx0010xx00xxx00000000000000000)

Bit	Reset	Description
27	0x0	CMD_DQS_TX_DRVFORCEON: [PMC]
26	0x0	CMD_DQ_TX_DRVFORCEON: [PMC]
25	0x1	CMD_CMD_TX_DRVFORCEON: [PMC]
24	0x0	CMD_CMD_TX_E_DCC: [PMC]
21	0x0	CMD_DQSN_TX_E_WKPU: [PMC] Active high to enable weak pull up
20	0x0	CMD_DQSN_TX_E_WKPD: [PMC] Active high to enable weak pull down

Bit	Reset	Description
16	0x0	CMD_DQSN_TX_E_DCC: [PMC] Active high to enable DQSN TX Duty Cycle Correction Circuit
15:14	0x0	CMD_DQS_TX_EBOOST_PU_MODE: [PMC] Select DQ pull up pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
13	0x0	CMD_DQSP_TX_E_WKPU: [PMC] Active high to enable weak pull up
12	0x0	CMD_DQSP_TX_E_WKPD: [PMC] Active high to enable weak pull down
11:10	0x0	CMD_DQS_TX_EBOOST_PD_MODE: [PMC] Select DQ pull down pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
9	0x0	CMD_DQSP_TX_E_DCC: [PMC] Active high to enable DQSP TX Duty Cycle Correction Circuit
8	0x0	CMD_DQS_E_IVREF: [PMC] Enable clock internal VREF resistor ladder for strobe IO_CKP/N. 0 = disable, 1 = enable; wake up latency to be 30uS
7:6	0x0	CMD_DQ_TX_EBOOST_PU_MODE: [PMC] Select DQ pull up pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
5	0x0	CMD_DQ_TX_E_WKPU: [PMC] Active high to enable weak pull up
4	0x0	CMD_DQ_TX_E_WKPD: [PMC] Active high to enable weak pull down
3:2	0x0	CMD_DQ_TX_EBOOST_PD_MODE: [PMC] Select DQ pull down pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
1	0x0	CMD_DQ_TX_E_DCC: [PMC] Active high to enable DQ TX Duty Cycle Correction Circuit
0	0x0	CMD_DQ_E_IVREF: [PMC] Enable data internal VREF resistor ladder for data I/O[8:0]. 0 = disable, 1 = enable; wake up latency to be 30uS

18.11.2.488 EMC_PMACRO_DATA_PAD_TX_CTRL_0

OB control for data pad

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc64 | Read/Write: R/W | Reset: 0x02000000 (0bxxxx0010xx00xxx00000000000000000)

Bit	Reset	Description
27	0x0	DATA_DQS_TX_DRVFORCEON: [PMC]
26	0x0	DATA_DQ_TX_DRVFORCEON: [PMC]
25	0x1	DATA_CMD_TX_DRVFORCEON: [PMC]
24	0x0	DATA_CMD_TX_E_DCC: [PMC]
21	0x0	DATA_DQSN_TX_E_WKPU: [PMC] Active high to enable weak pull up
20	0x0	DATA_DQSN_TX_E_WKPD: [PMC] Active high to enable weak pull down
16	0x0	DATA_DQSN_TX_E_DCC: [PMC] Active high to enable DQSN TX Duty Cycle Correction Circuit
15:14	0x0	DATA_DQS_TX_EBOOST_PU_MODE: [PMC] Select DQ pull up pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
13	0x0	DATA_DQSP_TX_E_WKPU: [PMC] Active high to enable weak pull up
12	0x0	DATA_DQSP_TX_E_WKPD: [PMC] Active high to enable weak pull down
11:10	0x0	DATA_DQS_TX_EBOOST_PD_MODE: [PMC] Select DQ pull down pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
9	0x0	DATA_DQSP_TX_E_DCC: [PMC] Active high to enable DQSP TX Duty Cycle Correction Circuit
8	0x0	DATA_DQS_E_IVREF: [PMC] Enable clock internal VREF resistor ladder for strobe IO_CKP/N. 0 = disable, 1 = enable; wake up latency to be 30uS
7:6	0x0	DATA_DQ_TX_EBOOST_PU_MODE: [PMC] Select DQ pull up pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
5	0x0	DATA_DQ_TX_E_WKPU: [PMC] Active high to enable weak pull up
4	0x0	DATA_DQ_TX_E_WKPD: [PMC] Active high to enable weak pull down

Bit	Reset	Description
3:2	0x0	DATA_DQ_TX_EBOOST_PD_MODE: [PMC] Select DQ pull down pre-emphasis mode. 00 = disable, 01 = 1X edge boosting, 10 = 2X edge boosting, 11 = 4X edge boosting
1	0x0	DATA_DQ_TX_E_DCC: [PMC] Active high to enable DQ TX Duty Cycle Correction Circuit
0	0x0	DATA_DQ_E_IVREF: [PMC] Enable data internal VREF resistor ladder for data I/O[8:0]. 0 = disable, 1 = enable; wake up latency to be 30uS

18.11.2.489 EMC_PMACRO_COMMON_PAD_TX_CTRL_0

OB control for Bricks

Offset: 0xc68 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	0x1	CMD_DQS_TX_BPSDYNEN: [PMC]
2	0x1	CMD_DQ_TX_BPSDYNEN: [PMC]
1	0x1	DATA_DQS_TX_BPSDYNEN: [PMC]
0	0x1	DATA_DQ_TX_BPSDYNEN: [PMC]

18.11.2.490 EMC_PMACRO_DQ_TX_DRV_0

Transmit drive strength control register for DQ brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc70 | Read/Write: R/W | Reset: 0x1f1f1f1f (0bxx011111xx011111xx011111xx011111)

Bit	Reset	Description
29:24	0x1f	DATA_DQS_TX_DRVDN: [PMC] TX pull-down drive strength code for DQS.
21:16	0x1f	DATA_DQS_TX_DRVUP: [PMC] TX pull-up drive strength code for DQS.
13:8	0x1f	DATA_DQ_TX_DRVDN: [PMC] TX pull-down drive strength code for DQ.
5:0	0x1f	DATA_DQ_TX_DRVUP: [PMC] TX pull-up drive strength code for DQ.

18.11.2.491 EMC_PMACRO_CA_TX_DRV_0

Transmit drive strength control register for CA brick

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc74 | Read/Write: R/W | Reset: 0x1f1f1f1f (0bxx011111xx011111xx011111xx011111)

Bit	Reset	Description
29:24	0x1f	CMD_DQS_TX_DRVDN: [PMC] TX pull-down drive strength code for CK.
21:16	0x1f	CMD_DQS_TX_DRVUP: [PMC] TX pull-up drive strength code for CK.
13:8	0x1f	CMD_DQ_TX_DRVDN: [PMC] TX pull-down drive strength code for CA.
5:0	0x1f	CMD_DQ_TX_DRVUP: [PMC] TX pull-up drive strength code for CA.

18.11.2.492 EMC_PMACRO_AUTOCAL_CFG_COMMON_0

Autocal padmacro register for controls that are shared across all IOBRICKs

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc78 | Read/Write: R/W | Reset: 0x00000810 (0bxxxxxxxxxxxxxxxx0xx001000xx010000)

Bit	Reset	Description
16	DISABLE	E_CAL_BYPASS_DVFS: [PMC] If enabled, asserts E_CAL_BYPASS for all pad macros. To be used during DVFS only 0 = DISABLE 1 = ENABLE
13:8	0x8	E_CAL_UPDATE_HIGH: [PMC] Number of EMC clocks for which E_CAL_UPDATE pulse is asserted for CK pad
5:0	0x10	E_CAL_UPDATE_DELAY: [PMC] Wait time in EMC clock cycles between CK pad cal code change and E_CAL_UPDATE assertion

18.11.2.493 EMC_PMACRO_DDLLCAL_CAL_0

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

DDLLCAL_CAL is used by the DDLL master to communicate the calibration value to the DDLL slaves

Offset: 0xce0 | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxxxx00001000000)

Bit	Reset	Description
10:0	0x40	DDLLCAL_CAL: DDLL Calibration value (how many trimmer taps == 1ui)

18.11.2.494 EMC_PMACRO_DDLL_OFFSET_0

DDLL_OFFSET is used to verify margins of the trained DDLL values.

Offset: 0xce4 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xxx000000000xx000000)

Bit	Reset	Description
28	0x0	CMD_DDLL_OFFSET_BYTE_QU_EN: Enable DDLL offset
27	0x0	CMD_DDLL_OFFSET_BYTE_RXDQS_EN: Enable DDLL offset
26	0x0	CMD_DDLL_OFFSET_BYTE_TXCMD_EN: Enable DDLL offset
25	0x0	CMD_DDLL_OFFSET_BYTE_TXDQS_EN: Enable DDLL offset
24	0x0	CMD_DDLL_OFFSET_BYTE_TXDQ_EN: Enable DDLL offset
23	0x0	CMD_DDLL_OFFSET_BIT_RXDQ_EN: Enable DDLL offset
22	0x0	CMD_DDLL_OFFSET_BIT_TXCMD_EN: Enable DDLL offset
21	0x0	CMD_DDLL_OFFSET_BIT_TXDQS_EN: Enable DDLL offset
20	0x0	CMD_DDLL_OFFSET_BIT_TXDQ_EN: Enable DDLL offset
16	0x0	DATA_DDLL_OFFSET_BYTE_QU_EN: Enable DDLL offset
15	0x0	DATA_DDLL_OFFSET_BYTE_RXDQS_EN: Enable DDLL offset
14	0x0	DATA_DDLL_OFFSET_BYTE_TXCMD_EN: Enable DDLL offset
13	0x0	DATA_DDLL_OFFSET_BYTE_TXDQS_EN: Enable DDLL offset
12	0x0	DATA_DDLL_OFFSET_BYTE_TXDQ_EN: Enable DDLL offset
11	0x0	DATA_DDLL_OFFSET_BIT_RXDQ_EN: Enable DDLL offset
10	0x0	DATA_DDLL_OFFSET_BIT_TXCMD_EN: Enable DDLL offset
9	0x0	DATA_DDLL_OFFSET_BIT_TXDQS_EN: Enable DDLL offset

Bit	Reset	Description
8	0x0	DATA_DDLL_OFFSET_BIT_TXDQ_EN: Enable DDLL offset
5:0	0x0	DDLL_OFFSET: Signed Offset to apply to each DDLL

18.11.2.495 EMC_PMACRO_DDLL_PERIODIC_OFFSET_0

DDLL_PERIODIC_OFFSET is used to support periodic training

Offset: 0xce8 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xxx000000000xx000000)

Bit	Reset	Description
28	0x0	CMD_DDLL_PERIODIC_OFFSET_BYTE_QU_EN: Enable DDLL offset
27	0x0	CMD_DDLL_PERIODIC_OFFSET_BYTE_RXDQS_EN: Enable DDLL offset
26	0x0	CMD_DDLL_PERIODIC_OFFSET_BYTE_TXCMD_EN: Enable DDLL offset
25	0x0	CMD_DDLL_PERIODIC_OFFSET_BYTE_TXDQS_EN: Enable DDLL offset
24	0x0	CMD_DDLL_PERIODIC_OFFSET_BYTE_TXDQ_EN: Enable DDLL offset
23	0x0	CMD_DDLL_PERIODIC_OFFSET_BIT_RXDQ_EN: Enable DDLL offset
22	0x0	CMD_DDLL_PERIODIC_OFFSET_BIT_TXCMD_EN: Enable DDLL offset
21	0x0	CMD_DDLL_PERIODIC_OFFSET_BIT_TXDQS_EN: Enable DDLL offset
20	0x0	CMD_DDLL_PERIODIC_OFFSET_BIT_TXDQ_EN: Enable DDLL offset
16	0x0	DATA_DDLL_PERIODIC_OFFSET_BYTE_QU_EN: Enable DDLL offset
15	0x0	DATA_DDLL_PERIODIC_OFFSET_BYTE_RXDQS_EN: Enable DDLL offset
14	0x0	DATA_DDLL_PERIODIC_OFFSET_BYTE_TXCMD_EN: Enable DDLL offset
13	0x0	DATA_DDLL_PERIODIC_OFFSET_BYTE_TXDQS_EN: Enable DDLL offset
12	0x0	DATA_DDLL_PERIODIC_OFFSET_BYTE_TXDQ_EN: Enable DDLL offset
11	0x0	DATA_DDLL_PERIODIC_OFFSET_BIT_RXDQ_EN: Enable DDLL offset
10	0x0	DATA_DDLL_PERIODIC_OFFSET_BIT_TXCMD_EN: Enable DDLL offset
9	0x0	DATA_DDLL_PERIODIC_OFFSET_BIT_TXDQS_EN: Enable DDLL offset
8	0x0	DATA_DDLL_PERIODIC_OFFSET_BIT_TXDQ_EN: Enable DDLL offset
5:0	0x0	DDLL_PERIODIC_OFFSET: Signed Offset to apply to each DDLL

18.11.2.496 EMC_PMACRO_VTTGEN_CTRL_2_0

VTTGEN control register

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xcf0 | Read/Write: R/W | Reset: 0x00100404 (0bxxxxxxxx00010000000010000000100)

Bit	Reset	Description
23:16	0x10	VTT_VDDA_LOAD: [PMC]
15:8	0x4	VTT_VAUXP_LOAD: [PMC]
7:0	0x4	VTT_VCLAMP_LOAD: [PMC]

18.11.2.497 EMC_PMACRO_IB_RXRT_0

IB DQ Receive path retiming for Byte0 to Byte3

This register is shadowed: see usage notes in Section 18.11.2: EMC Registers.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xcf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx00000000000)

Bit	Reset	Description
10:0	0x0	IB_RXRT: [PMC] This should be programmed to IB DQ Receive path retiming

18.11.2.498 EMC_PMACRO_TRAINING_CTRL_0_0

Used by training engine to configuration pad macros for Channel 0

Offset: 0xcf8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	CH0_TRAINING_DRV_DQS: When training is enabled, force the DQS_t and DQS_c drivers on
3	0x0	CH0_TRAINING_E_WRPTR: Enables direct return of the ififo wrptr as read data if training is enabled
2	0x0	CH0_TRAINING_RX_E_DIRECT_ZI: Training override for the RX_E_DIRECT_ZI pin of the pad
1	0x0	CH0_TRAINING_TRAIN_QPOP: Set to 1 when training QPOP
0	0x0	CH0_TRAINING_ENABLED: Indicates training is enabled for this channel

18.11.2.499 EMC_PMACRO_TRAINING_CTRL_1_0

Used by training engine to configuration pad macros for Channel 1

Offset: 0xcfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	CH1_TRAINING_DRV_DQS: When training is enabled, force the DQS_t and DQS_c drivers on
3	0x0	CH1_TRAINING_E_WRPTR: Enables direct return of the ififo wrptr as read data if training is enabled
2	0x0	CH1_TRAINING_RX_E_DIRECT_ZI: Training override for the RX_E_DIRECT_ZI pin of the pad
1	0x0	CH1_TRAINING_TRAIN_QPOP: Set to 1 when training QPOP
0	0x0	CH1_TRAINING_ENABLED: Indicates training is enabled for this channel

18.11.2.500 EMC_TRAINING_CMD_0

Main register for launching training

Offset: 0xe00 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31	DISABLED	GO: Start the Training. Training will automatically stop. This bit is cleared by hardware when Training is done. Refer to the Training status register definition. 0 = DISABLED 1 = ENABLED
30	DISABLED	PERIODIC: Start the periodic Training 0 = DISABLED 1 = ENABLED
8	DISABLED	QUSE_VREF:Initiates DQS_VREF Training 0 = DISABLED 1 = ENABLED
7	DISABLED	RD_VREF:Initiates IB_DQ_VREF Training 0 = DISABLED 1 = ENABLED
6	DISABLED	WR_VREF:Initiates OB (wrire) DRAM_VREF Training 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
5	DISABLED	CA_VREF:Initiates CA_VREF Training 0 = DISABLED 1 = ENABLED
4	DISABLED	QUSE:Initiates QUSE Training 0 = DISABLED 1 = ENABLED
3	DISABLED	WR:Initiates WR Training 0 = DISABLED 1 = ENABLED
2	DISABLED	RD:Initiates RD Training 0 = DISABLED 1 = ENABLED
1	DISABLED	CA:Initiates CA Training 0 = DISABLED 1 = ENABLED
0	DISABLED	PRIME:Writes the custom pattern into MPC register 0 = DISABLED 1 = ENABLED

18.11.2.501 EMC_TRAINING_CTRL_0

MISC controls for training configuration

Offset: 0xe04 | Read/Write: R/W | Reset: 0x00009080 (0bxxxxxxxxxxxxxxxx01001000010000000)

Bit	Reset	Description
16	DISABLED	FORCE_DR_TO_SR:When config is dual rank - only RANK0 will be swept and Training engine will stop and not do RANK1 sweep. 0 = DISABLED 1 = ENABLED
15	ENABLED	TR_IN_SELF_REFRESH:Selects if Training is done in self refresh 0 = DISABLED 1 = ENABLED

14	RANK0	SWAP_RANK:FORCE_DR_TO_SR is set - denotes - which rank should be trained. If FORCE_DR_TO_SR = 0 for dual rank - Denotes the order in which ranks will be trained //if RANK0 : RANK0_followed_by_RANK1. if RANK1 - RANK1_Followed_by_RANK0.
		0 = RANK0
		1 = RANK1

13	DISABLED	UPDATE_QUSE_RD_TR:Selects if QUSE PRIV needs to be updated during read training. 0 = DISABLED 1 = ENABLED
12	IFIFO_WRPTR	QUSE_MODE: Selects between expect Actual read data or IFIFO WRPTR for QUSE Training read returns 0 = READ_DATA 1 = IFIFO_WRPTR
11:4	0x8	IFIFO_WRPTR: Expected WRPTR value for expected read_data in QUSE Training when RPT mode is unavailable in DRAM
3	DISABLED	RPT_MODE: Indicates read preamble training mode is available in DRAM. 0 = DISABLED 1 = ENABLED
2	DISABLED	REFRESH_CA:allows refreshes in CA Training 0 = DISABLED 1 = ENABLED
1	DISABLED	REFRESH:allows refreshes in all Training but CA Training. 0 = DISABLED 1 = ENABLED
0	DISABLED	ASYNC_UPDATES: This bit must be disabled. 0 = DISABLED 1 = ENABLED

18.11.2.502 EMC_TRAINING_STATUS_0

Read back for Training status

Offset: 0xe08 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	STATUS:Writes 0 = DONE 1 = RUNNING 2 = ERROR

18.11.2.503 EMC_TRAINING_QUSE_CORS_CTRL_0

Controls for QUSE TRainings cors sweep

Offset: 0xe0c | Read/Write: R/W | Reset: 0x0111e000 (0bxxx000010001x001111000x000000000)

Bit	Reset	Description
28	ZERO	STEP_LN_SIZE: 0 = ZERO 1 = ONE
27:24	ONE	STEP_LN: 0 = ZERO 1 = ONE
23:20	ONE	STEP: 0 = ZERO 1 = ONE
18:10	PROD	MAXIMUM: 120 = PROD
8:0	ZERO	MIMIMUM: 0 = ZERO

18.11.2.504 EMC_TRAINING_QUSE_FINE_CTRL_0

Controls for QUSE Trainings fine sweep

Offset: 0xe10 | Read/Write: R/W | Reset: 0x1110ffc1 (0bxxx10001000100001111111111000001)

Bit	Reset	Description
28	ONE	STEP_LN_SIZE: 0 = ZERO 1 = ONE
27:24	ONE	STEP_LN: 0 = ZERO 1 = ONE
23:20	ONE	STEP: 0 = ZERO 1 = ONE
19:10	PLUS_63	MAXIMIM: 2's complement 63 = PLUS_63
9:0	MINUS_63	MIMIMUM:2's complement 961 = MINUS_63

18.11.2.505 EMC_TRAINING_QUSE_CTRL_MISC_0

MISC Controls for QUSE Training

Offset: 0xe14 | Read/Write: R/W | Reset: 0x15081000 (0b00010101000010000001000000000000)

Bit	Reset	Description
31:24	0x15	PATRAM_MAX:End index in patram. Program as PATRAM_MIN + (2* BURST_COUNT)-1

Bit	Reset	Description
23:16	0x8	PATRAM_MIN:Starting index in patram
15:14	BYTE	LEVEL:Selects between byte vs bit level training style 0 = BYTE 1 = BIT
13:12	MPC	PATTERN:Selects between MPC register or training FIFO 0 = FIFO 1 = MPC
11:8	ONE	BURST_COUNT:Number of reads in each STEP_LN. Number of reads each step_ln = BURST_COUNT + 1 0 = ONE 1 = TWO 2 = THREE 3 = FOUR
7:0	ZERO	ERR_LIMIT:Number of errors allowed before considered a failure 0 = ZERO

18.11.2.506 EMC_TRAINING_WRITE_FINE_CTRL_0

Controls for Write Trainings sweep

Offset: 0xe18 | Read/Write: R/W | Reset: 0x11150000 (0bxxx10001000101010000000000000000)

Bit	Reset	Description
28	ONE	STEP_LN_SIZE: 0 = ZERO 1 = ONE
27:24	ONE	STEP_LN: 0 = ZERO 1 = ONE
23:20	ONE	STEP: 0 = ZERO 1 = ONE
19:10	PROD	MAXIMUM: 320 = PROD
9:0	ZERO	MIMIMUM: 0 = ZERO

18.11.2.507 EMC_TRAINING_WRITE_CTRL_MISC_0

MISC Controls for Write Trainings sweep

Offset: 0xe1c | Read/Write: R/W | Reset: 0x07000300 (0b00000111000000000000001100000000)

Bit	Reset	Description
31:24	0x7	PATRAM_MAX:End index in patram. Program as Always PATRAM_MIN + (2* BURST_COUNT)-1
23:16	0x0	PATRAM_MIN
15:14	BYTE	LEVEL: 0 = BYTE 1 = BIT
13:12	FIFO	PATTERN: 0 = FIFO 1 = MPC
11:8	FOUR	BURST_COUNT:Number of reads in each STEP_LN. Number of reads each step_ln = BURST_COUNT + 1 0 = ONE 1 = TWO 2 = THREE 3 = FOUR
7:0	ZERO	ERR_LIMIT: 0 = ZERO

18.11.2.508 EMC_TRAINING_WRITE_VREF_CTRL_0

Controls for Write-vref Training. Based on OB(write) is terminated or unterminated: Ctrl will need to be changed. Default controls are select for terminated case

Offset: 0xe20 | Read/Write: R/W | Reset: 0x00103200 (0bxxxxxxxx0001xxxx0011001000000000)

Bit	Reset	Description
23:20	ONE	STEP: 0 = ZERO 1 = ONE
15:8	PROD	MAXIMUM: 50 = PROD
7:0	ZERO	MIMIMUM: 0 = ZERO

18.11.2.509 EMC_TRAINING_READ_FINE_CTRL_0

Controls for READ Trainings sweep

Offset: 0xe24 | Read/Write: R/W | Reset: 0x1110fc00 (0bxxx10001000100001111110000000000)

Bit	Reset	Description
28	ONE	STEP_LN_SIZE: 0 = ZERO 1 = ONE
27:24	ONE	STEP_LN: 0 = ZERO 1 = ONE
23:20	ONE	STEP: 0 = ZERO 1 = ONE
19:10	PROD	MAXIMUM: 63 = PROD
9:0	ZERO	MIMIMUM: 0 = ZERO

18.11.2.510 EMC_TRAINING_READ_CTRL_MISC_0

MISC Controls for Read Trainings sweep

Offset: 0xe28 | Read/Write: R/W | Reset: 0x15081300 (0b00010101000010000001001100000000)

Bit	Reset	Description
31:24	0x15	PATRAM_MAX:End index in patram. Program as Always PATRAM_MIN + (2* BURST_COUNT)-1
23:16	0x8	PATRAM_MIN:Start index in patram
15:14	BYTE	LEVEL: 0 = BYTE 1 = BIT
13:12	MPC	PATTERN: 0 = FIFO 1 = MPC
11:8	FOUR	BURST_COUNT:Number of reads in each STEP_LN. Number of reads each step_ln = BURST_COUNT + 1 3 = FOUR
7:0	ZERO	ERR_LIMIT: 0 = ZERO

18.11.2.511 EMC_TRAINING_READ_VREF_CTRL_0

Controls for Read-vref Trainings sweep

Offset: 0xe2c | Read/Write: R/W | Reset: 0x00103f00 (0bxxxxxxxx0001xxxx0011111100000000)

Bit	Reset	Description
23:20	ONE	STEP: 0 = ZERO 1 = ONE
15:8	PROD	MAXIMUM: 63 = PROD
7:0	ZERO	MIMIMUM: 0 = ZERO

18.11.2.512 EMC_TRAINING_CA_FINE_CTRL_0

Controls for CA Training sweep

Offset: 0xe30 | Read/Write: R/W | Reset: 0x11130030 (0bxxx10001000100110000000000110000)

Bit	Reset	Description
28	ONE	STEP_LN_SIZE: 0 = ZERO 1 = ONE
27:24	ONE	STEP_LN: 0 = ZERO 1 = ONE
23:20	ONE	STEP: 0 = ZERO 1 = ONE
19:10	PROD	MAXIMUM: 192 = PROD
9:0	PROD	MIMIMUM: 48 = PROD

18.11.2.513 EMC_TRAINING_CA_CTRL_MISC_0

MISC Controls for CA Trainings sweep.

Offset: 0xe34 | Read/Write: R/W | Reset: 0x1f101300 (0b00011111000010000000010011000000000)

Bit	Reset	Description
31:24	0x1f	PATRAM_MAX
23:16	0x10	PATRAM_MIN
15:14	BYTE	LEVEL: 0 = BYTE 1 = BIT
13:12	MPC	PATTERN:Unusded for Ca Training. CATR_Rd are used whcih are special. 0 = FIFO 1 = MPC
11:8	FOUR	BURST_COUNT:unused for Ca Training 3 = FOUR
7:0	ZERO	ERR_LIMIT: 0 = ZERO

18.11.2.514 EMC_TRAINING_CA_CTRL_MISC1_0

Misc Controls for CA Training sweep

Offset: 0xe38 | Read/Write: R/W | Reset: 0x00000018 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00011000)

Bit	Reset	Description
7:0	PROD	SKIP_CNT:number of NOPs between 2 CA Training reads. Number of NOP's @ DRAMCLK = (SKIP_CNT * 2 * MCCLK_PERIOD/EMCCLK_PERIOD) + (2 * MCCLK_PERIOD/EMCCLK_PERIOD) - 1 24 = PROD

18.11.2.515 EMC_TRAINING_CA_VREF_CTRL_0

Controls for CA-VREF Training

Offset: 0xe3c | Read/Write: R/W | Reset: 0x00103200 (0bxxxxxxxx0001xxxx0011001000000000)

Bit	Reset	Description
23:20	ONE	STEP: 0 = ZERO 1 = ONE
15:8	PROD	MAXIMUM: 50 = PROD
7:0	PROD	MIMIMUM: 0 = PROD

18.11.2.516 EMC_TRAINING_CA_TADR_CTRL_0

Controls for TADR Sweep

Offset: 0xe40 | Read/Write: R/W | Reset: 0x00028000 (0b0xxxxxxxxxxx00101000000000000000)

Bit	Reset	Description
31	DISABLED	TADR_ENABLE: This bit must be disabled. 0 = DISABLED 1 = ENABLED
19:12	PROD	TADR_MAX: maximum tADR allowed in tCK. 40 = PROD
11:0	ZERO	TADR_SETTING: CA BRLSHFT + CA trimmer during tADR Training 0 = ZERO

18.11.2.517 EMC_TRAINING_SETTLE_0

Controls for settle time for delay controls in MCCLK

Offset: 0xe44 | Read/Write: R/W | Reset: 0x07070404 (0b00000111000001110000010000000100)

Bit	Reset	Description
31:24	0x7	APPLY_DELAY:wait time after issuing a training priv request
23:16	0x7	PRIV_DELAY:wait time after issuing a training priv request
15:8	0x4	SHORT_TRIMMER: Settle time for short trimmer
7:0	0x4	LONG_TRIMMER: Settle time for long trimmer

18.11.2.518 EMC_TRAINING_DEBUG_CTRL_0

Controls to select which debug information should be output in DEBUG_DQ* registers

Offset: 0xe48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
27:24	0x0	WINDOW_LIMIT: threshold to determine how many windows to report in debug registers. window is reported in window count if window_size > window_limit

Bit	Reset	Description
7:0	0x0	SELECT: specifies what is output on DQ registers 80 = RANK0_CA_PASSING0 81 = RANK0_CA_PASSING1 82 = RANK0_CA_PASSING2 83 = RANK0_CA_PASSING3 84 = RANK0_CA_PASSING_BYTE 85 = RANK1_CA_PASSING0 86 = RANK1_CA_PASSING1 87 = RANK1_CA_PASSING2 88 = RANK1_CA_PASSING3 89 = RANK1_CA_PASSING_BYTE 11 = RANK0_DQ_OB_PASSING0 12 = RANK0_DQ_OB_PASSING1 13 = RANK0_DQ_OB_PASSING2 14 = RANK0_DQ_OB_PASSING3 15 = RANK0_DQ_OB_PASSING4 16 = RANK0_DQ_OB_PASSING5 17 = RANK0_DQ_OB_PASSING6 18 = RANK0_DQ_OB_PASSING7 19 = RANK0_DQ_OB_PASSING_DBI 21 = RANK0_DQ_OB_PASSING_BYTE 0 = RANK0_DQ_IB_PASSING0 1 = RANK0_DQ_IB_PASSING1 2 = RANK0_DQ_IB_PASSING2 3 = RANK0_DQ_IB_PASSING3 4 = RANK0_DQ_IB_PASSING4 5 = RANK0_DQ_IB_PASSING5 6 = RANK0_DQ_IB_PASSING6 7 = RANK0_DQ_IB_PASSING7 8 = RANK0_DQ_IB_PASSING_DBI 10 = RANK0_DQ_IB_PASSING_BYTE 155 = RANK1_DQ_OB_PASSING0 156 = RANK1_DQ_OB_PASSING1 157 = RANK1_DQ_OB_PASSING2 158 = RANK1_DQ_OB_PASSING3 159 = RANK1_DQ_OB_PASSING4 160 = RANK1_DQ_OB_PASSING5 161 = RANK1_DQ_OB_PASSING6 162 = RANK1_DQ_OB_PASSING7 163 = RANK1_DQ_OB_PASSING_DBI 165 = RANK1_DQ_OB_PASSING_BYTE 144 = RANK1_DQ_IB_PASSING0 145 = RANK1_DQ_IB_PASSING1 146 = RANK1_DQ_IB_PASSING2 147 = RANK1_DQ_IB_PASSING3 148 = RANK1_DQ_IB_PASSING4 149 = RANK1_DQ_IB_PASSING5 150 = RANK1_DQ_IB_PASSING6 151 = RANK1_DQ_IB_PASSING7 152 = RANK1_DQ_IB_PASSING_DBI 154 = RANK1_DQ_IB_PASSING_BYTE 203 = RANK0_DQ_QUSE_PASSING0 204 = RANK0_DQ_QUSE_PASSING1 205 = RANK0_DQ_QUSE_PASSING2 206 = RANK0_DQ_QUSE_PASSING3 207 = RANK0_DQ_QUSE_PASSING4 208 = RANK0_DQ_QUSE_PASSING5 209 = RANK0_DQ_QUSE_PASSING6 210 = RANK0_DQ_QUSE_PASSING7 211 = RANK0_DQ_QUSE_PASSING_DBI 213 = RANK0_DQ_QUSE_PASSING_BYTE 176 = RANK1_DQ_QUSE_PASSING0 177 = RANK1_DQ_QUSE_PASSING1 178 = RANK1_DQ_QUSE_PASSING2 179 = RANK1_DQ_QUSE_PASSING3 180 = RANK1_DQ_QUSE_PASSING4 181 = RANK1_DQ_QUSE_PASSING5 182 = RANK1_DQ_QUSE_PASSING6 183 = RANK1_DQ_QUSE_PASSING7 184 = RANK1_DQ_QUSE_PASSING_DBI 186 = RANK1_DQ_QUSE_PASSING_BYTE 96 = DQ_ERRCNT 128 = SEQUENCER 129 = INTERFACE

18.11.2.519 EMC_TRAINING_MPC_0

16-bit data for MPC-RD registers. DRAM stores this information in MR32 and MR40. 16-bit invert pattern for DQ. DRAM stores this information in MR12 and MR15. When using Default MPC pattern for read/Quse - Write this register with default pattern. when using custom pattern for readquse - Write this register along with MR32, MR40, M15, M20. Writing to this register triggers an automated patram initialization of some pf the patram location. Pointed by QUSE_MISC_PATRAM_MIN/MAX

Offset: 0xe5c | Read/Write: R/W | Reset: 0x05053c5a (0b00000101000001010011110001011010)

Bit	Reset	Description
31:16	0x505	INVERT_PATTERN
15:0	0x3c5a	DATA

18.11.2.520 EMC_TRAINING_PATRAM_CTRL_0

Register to write pattern RAM

Offset: 0xe60 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxx0000xxxxxxxx00000000)

Bit	Reset	Description
31	0x0	WRITE:when write=1 - trigger a write into pattern RAM with data supplied by PATRAM_DATA_* registers
19:16	0x0	FORMAT
7:0	0x0	SELECT_OFFSET: locationt to write in pattern RAM

18.11.2.521 EMC_TRAINING_PATRAM_DQ_0

DQ data to write into pattern RAM

Offset: 0xe64 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA: locationt to write in pattern RAM

18.11.2.522 EMC_TRAINING_PATRAM_DMI_0

DMI data to write into pattern RAM

Offset: 0xe68 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA: locationt to write in pattern RAM

18.11.2.523 EMC_TRAINING_VREF_SETTLE_0

Controls for settle time for vref in MCCLK

Offset: 0xe6c | Read/Write: R/W | Reset: 0x000007d0 (0b00000000000000000000011111010000)

Bit	Reset	Description
31:16	0x0	OB: Settle time for WRITE/CA vrefs. From DRAM specification. CA/OB receiver settle time is 200ns with VRCG mode(MR13-bit3) enabled. With VRCG disabled 1us. Training engine does not implement this delay.
15:0	0x7d0	IB: Settle time for READ/QUSE vrefs. For circuit folks - AP's receiver vref settle time is 1us. Do not program PRIV_DELAY + VREF_SETTLE to be more than a 16b number. 16b number gives far more than required range.

18.11.2.524 EMC_TRAINING_OPT_CA_VREF_0

Read back for CA VREF optimal Vref results

Offset: 0xec0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	RANK1_SUB_PARTITION1:CA_VREF optimal for RANK1 SUB_PARTITION1 (16b) for each channel. For single rank readback is zero.
23:16	X	RANK1_SUB_PARTITION0:CA_VREF optimal for RANK1 SUB_PARTITION0 (16b) for each Channel. For single rank readback is zero.
15:8	X	RANK0_SUB_PARTITION1:CA_VREF optimal for RANK0 SUB_PARTITION1(16b) for each Channel
7:0	X	RANK0_SUB_PARTITION0:CA_VREF optimal for RANK0 SUB_PARTITION0(16b) for each Channel

18.11.2.525 EMC_TRAINING_OPT_DQ_OB_VREF_0

Read back for OB-DQ VREF optimal Vref results

Offset: 0xec4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	RANK1_SUB_PARTITION1:DQ_OB_VREF optimal for RANK1 SUB_PARTITION1(16b) for each Channel. For single rank readback is zero.
23:16	X	RANK1_SUB_PARTITION0:DQ_OB_VREF optimal for RANK1 SUB_PARTITION0(16b) for each Channel. For single rank readback is zero.
15:8	X	RANK0_SUB_PARTITION1:DQ_OB_VREF optimal for RANK0 SUB_PARTITION1(16b) for each Channel
7:0	X	RANK0_SUB_PARTITION0:DQ_OB_VREF optimal for RANK0 SUB_PARTITION0(16b) for each Channel

18.11.2.526 EMC_TRAINING_QUSE_VREF_CTRL_0

Controls for Read-DQS-Vref trainings sweep.

Offset: 0xed0 | Read/Write: R/W | Reset: 0x00103f00 (0bxxxxxxxx0001xxxx0011111100000000)

Bit	Reset	Description
23:20	ONE	STEP: 0 = ZERO 1 = ONE
15:8	PROD	MAXIMUM: 63 = PROD
7:0	ZERO	MIMIMUM: 0 = ZERO

CHAPTER 19: AHB

19.1 AHB Bus

The AHB Bus conforms to the *AMBA® Specification (Rev 2.0) Advanced High-performance Bus (AHB)* architecture as published by ARM.

AHB is a 32-bit multi-master bus. Despite what its name implies, it is a second tier bus in the Tegra® X1 processor, slower and less flexible than the AXI bus used by the CPU, or the memory client interface used by the high-speed devices.

The Master clients on the AHB are:

- The AHB memory controller slave, which is the interface between the AHB bus and the Memory Controller
- The BPMP-Lite crossbar, both as a master and as a slave. This is the path to IRAM from AHB devices, and also the BPMP-Lite and CPU path to AHB devices. Neither the CPU complex nor the BPMP-Lite can access DRAM through this path.
- The APB DMA controller, used to provide data transfer between APB devices and memory
- The USB-OTG controller
- ARC controller
- The security engine (SE), both as a master and as a slave
- The deprecated AHB DMA controller master
- USB2 controllers

The Slave clients on the AHB are:

- XBAR
- DRAM
- USB-OTG, USB2
- TZRAM

AHB in the Tegra X1 processor supports secure access to memory, using the same secure bit mechanism used by the CPU TrustZone® security mechanism. This is supported by the SE, which can make secure accesses to memory.

19.2 AHB Bus Arbiter

The AHB Arbiter controls AHB bus master arbitration. This effectively forms a second level of arbitration for access to the memory controller through the AHB Slave Memory device, although AHB masters in some circumstances can use other AHB slaves, in particular they can access IRAM.

The controls presented here are largely for diagnostic purposes only. For suspect AHB performance problems, try disabling the other masters for the device with performance issues. In normal operation bus parking is usually enabled, and all the masters are enabled.

Also see the AHB slave interface registers, where the AHB pre-fetch logic can be configured to enhance performance for devices doing sequential read access from DRAM.

Each AHB master is assigned to either the high or low priority bin.

19.2.1 Arbitration Scheme

In every AHB Arbitration cycle, it is first decided whether a request from the high or low priority bin will be served. If there are only requests active for one of the bins, then that bin is served. When there are active requests from both bins, then the value from the AHB_PRIORITY_WEIGHT field is considered. There is a counter that is loaded with the AHB_PRIORITY_WEIGHT whenever the current count is 0 and someone wins an AHB arbitration. This counter is decremented anytime the count is nonzero and the winner of AHB arbitration was from the high-priority bin. Whenever this counter is nonzero, then a high-priority bin request will win over any low-priority bin request. In a system where both high- and low- priority bin requests are constantly active, the AHB_PRIORITY_WEIGHT says how many high-priority requests will be served for every one low-priority request.

Within each bin, the arbitration algorithm is round robin. For example, after AHB Master 2 wins an arbitration, then Master 3 has precedence for winning the next (followed by Master 4, etc. while Master 2 is last). The AHB Master ID can be seen in the enumeration of AHB_MEM_PREFETCH_CFG* registers' AHB_MST_ID field.

19.2.2 AHB Arbiter Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

In prior generation Tegra processors, this location used to hold a configuration register for COP_CACHE, but COP_CACHE has been replaced with AVP_CACHE, which has a region in a different section. For software compatibility, this location is reserved.

19.2.2.1 AHB_ARBITRATION_DISABLE_0

The AHB arbitration control register allows user to tweak arbitration behavior of the AHB arbiter.

- Enable bus parking. This keeps the last serviced AHB master on the bus granted so that it can start another transaction faster. If bus parking is disabled, no AHB master will be able to start a new transaction until the arbitration is done and the master is granted the bus.
- Allows the user to specifically disable an AHB master from arbitrating on the AHB bus.

AHB Arbitration Controller

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxx0x000xxxxx00000x00)

Bit	Reset	Description
31	0x0	DIS_BUS_PARK: 1 = Disable bus parking. 0 = ENABLE 1 = DISABLE
30	0x0	DIS_PENULTIMATE_ARB: 1 = Disable arbitration on the second to last transfer.
18	0x0	USB2: 1 = Disable USB2 from arbitration. 0 = ENABLE 1 = DISABLE
14	0x0	SE: 1 = Disable SE from arbitration. 0 = ENABLE 1 = DISABLE
7	0x0	APBDMA: 1 = Disable APB-DMA from arbitration. 0 = ENABLE 1 = DISABLE
6	0x0	USB: 1 = Disable USB from arbitration. 0 = ENABLE 1 = DISABLE
4	0x0	ARC: 1 = Disable ARC from arbitration. 0 = ENABLE 1 = DISABLE
1	0x0	COP: 1 = Disable COP from arbitration. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
0	0x0	CPU: 1 = Disable CPU from arbitration. 0 = ENABLE 1 = DISABLE

19.2.2.2 AHB_ARBITRATION_PRIORITY_CTRL_0

The AHB arbiter implements a 2-level priority scheme. In the first level, arbitration is determined between the high and low priority group according to the priority weight; the higher the weight, the higher the winning rate of the high priority group.

In the second level, within each of the high/low priority group, arbitration is determined in a round-robin fashion.

AHB Arbitration Priority Control Register

Offset: 0x8 | Read/Write: R/W | Reset: 0xe0000000 (0b11100000000000000000000000000000)

Bit	Reset	Description
31:29	0x7	AHB_PRIORITY_WEIGHT: AHB priority weight count. This 3-bit field is used to control the amount of attention (weight) given to the high priority group before switching to the low priority group.
28:0	0x0	AHB_PRIORITY_SELECT: 0 = low priority group.

19.2.2.3 AHB_ARBITRATION_USR_PROTECT_0

USR Protection Register

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000x0000)

Bit	Reset	Description
8	0x0	CACHE: Abort on USR mode access to Cache memory space 0 = ABT_DIS 1 = ABT_EN
7	0x0	ROM: Abort on USR mode access to internal ROM memory space 0 = ABT_DIS 1 = ABT_EN
6	0x0	APB: Abort on USR mode access to APB memory space 0 = ABT_DIS 1 = ABT_EN
5	0x0	AHB: Abort on USR mode access to AHB memory space 0 = ABT_DIS 1 = ABT_EN
3	0x0	IRAMD: Abort on USR mode access to iRAMd memory space 0 = ABT_DIS 1 = ABT_EN
2	0x0	IRAMC: Abort on USR mode access to iRAMc memory space 0 = ABT_DIS 1 = ABT_EN
1	0x0	IRAMB: Abort on USR mode access to iRAMb memory space 0 = ABT_DIS 1 = ABT_EN
0	0x0	IRAMA: Abort on USR mode access to iRAMa memory space 0 = ABT_DIS 1 = ABT_EN

19.3 AHB “Gizmo”

AHB master/slave gizmos are essentially hardware layers used by many of the master/slave logic which need to connect to the AHB bus.

These hardware layers handle all the AHB bus protocols and convert the more complex AHB protocol into a much simpler IP interface/handshake.

Because the AHB master/slave gizmos interface with many IP logic with different characteristics varying in speed, latency, etc., the gizmos accept a number of static configuration bits. Depending on system speed, latency requirement, transfer direction, burst characteristic, etc., these static configuration bits can be pre-configured to give extra performance or improve bus efficiency.

19.3.1 AHB Gizmo Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

19.3.1.1 AHB_GIZMO_AHB_MEM_0

AHB Master/Slave Gizmo Register

Given below is a description of each configuration field, what they are used for and the pros and cons of using them.

> ***Note:*** *These configuration bits are meant to be changed only when the particular gizmo you want to change is idle (not being used). Changing the configuration bits on-the-fly while that gizmo is active can hang the system.*

AHB gizmo configuration bit definition:

(1) AHB Master Gizmo

1. MAX_AHB_BURSTSIZE: Controls the maximum burst size that this gizmo can generate on the AHB bus. For the current system, this field should be set to burst-of-8.
2. IMMEDIATE: Controls how quickly an AHB write request on the bus can start.
   - If 1, the gizmo will start an AHB write request once the IP side provides one beat of data of a burst. For IP that can provide quick continuous burst data, this setting provides parallelism between AHB request and IP data transfer. However, for IP that cannot provide quick burst data, this setting will force this gizmo to hold the AHB bus longer, reducing bus efficiency.
   - If 0, gizmo will wait until all beats of data of a burst are provided before starting an AHB write request. With this setting, AHB bus efficiency is realized but IP latency and efficiency may be reduced.
3. RD_DATA: Controls how read data will be returned from the gizmo to the IP side.
   - If 1, all beats of data of a burst have to be available before it indicates to the IP logic that read data is ready. This setting ensures there is no wait-state (bubble) between read data burst, but it will increase latency.
   - If 0, each beat of data of a burst will be sent from gizmo to the IP logic immediately. This setting will reduce latency, but can create non-consecutive data burst or bubble between data.
4. REQ_NEG_CNT: Provides a way to limit (in terms of number of AHB bus clock) how fast/often this master can request the AHB bus. The bigger the number, the slower the rate of AHB request.

(2) AHB Slave Gizmo

1. ENABLE_SPLIT: Controls enabling the split feature of the AHB bus.
   - If 1, the gizmo will always generate split response for a read. If 'DONT_SPLIT_AHB_WR' is set to 0, gizmo also generates split response if it can no longer accept a write request from the AHB master. This setting can increase AHB bus utilization since it allows other AHB masters to talk to other AHB slaves while this original AHB slave is fetching read data. However, the flip side is that it takes longer time for the original AHB master to get the read data, because the original AHB master has to re-arbitrate for the AHB bus again to get to the data it asked for.
   - If 0, the gizmo will not generate split response. Instead, it will hold on to the AHB bus until all the requested read data is returned back to the AHB master. This setting will reduce read latency to the master, but decrease AHB bus utilization.
2. FORCE_TO_AHB_SINGLE: Controls how the gizmo treats the AHB master's burst request.
   - If 1, the gizmo will break up the burst request internally into individual single word requests.

- If 0, the gizmo will burst request internally as burst request.

3. ENB_FAST_REARBITRATE: Controls when the gizmo can allow the original read requested AHB master to re-arbitrate for the AHB bus again so the AHB master can retrieve the originally requested read data.
   - If 1, once first read data of a burst is in the slave gizmo's FIFO, it will allow the original AHB master to re-arbitrate, thus, allowing arbitration to happen in parallel with subsequent read data of a burst. However, if the data burst has wait-states or bubbles, then this setting will decrease AHB bus utilization because the slave gizmo will hold on to the AHB bus longer.
   - If 0, all read data of a burst must be in the slave gizmo's FIFO before it allows the original AHB master to re-arbitrate to retrieve the read data. This increases read data latency, and also increase AHB bus utilization.
4. IP_WR_REQ_IMMEDIATE: Controls when the gizmo will start write data request to the IP logic.
   - If 1, the gizmo will start write request to the IP logic once it gets one write data of a burst from the AHB side. This setting will create non-consecutive/bubbles in a write data burst.
   - If 0, the gizmo will start write request to the IP logic only when it gets all write data of a burst from the AHB side. This setting will create consecutive data burst (no bubbles or wait-states).
5. MAX_IP_BURSTSIZE: Controls the maximum burst size that this gizmo can generate to the IP logic.
6. ACCEPT_AHB_WR_ALWAYS: Controls how the slave gizmo will treat AHB write request.
   - If 1, the gizmo will always accept a write request without checking whether it's FIFOs can accept the write or not. This setting can reduce bus utilization, but can reduce the rate of AHB retry.
   - If 0, the gizmo will check its FIFOs to make sure they have room before accepting the AHB write request. This setting increase bus utilization, but can create a lot of AHB retry.
7. DONT_SPLIT_AHB_WR: Controls whether to split AHB write request when the slave gizmo determines it cannot accept the write request.
   - If 1 and when ENABLE_SPLIT=1, the gizmo will generate split for write if its FIFOs are not ready to accept the AHB write request or data. This setting can improve AHB bus utilization as there are no continuous AHB master retries on the bus.
   - If 0, the gizmo will generate retry response for write if its FIFOs are not ready to accept the AHB write request. Software should leave this bit at 0 since this feature has not been proven.

AHB Gizmo AHB-DMA/Memory Control Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x00020385 (0b00000000xxxxx010xxxxxx1110xxx101)

Bit	Reset	SW Default	Description
31:24	0x0	NONE	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	NONE	IMMEDIATE: AHB master gizmo (AHB-DMA) – Start AHB write request immediately 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	NONE	MAX_AHB_BURSTSIZE: AHB master gizmo (AHB-DMA) – Maximum allowed AHB burst size. 00 = Single transfer 01 = Burst-of-4 10 = Burst-of-8 11 = Burst-of-16 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

Bit	Reset	SW Default	Description
9	0x1	ENABLE	EN_USB_WAIT_COMMIT_ON_XK_STALL: Enabling this register and enabling WR_WAIT_COMMIT_ON_XK causes MC requests to be stalled when the current request from the USB is in a different XKB block with respect to the previous USB request. Note that XK = 4K for Tegra X1 devices. 0 = DISABLE 1 = ENABLE
8	0x1	ENABLE	WR_WAIT_COMMIT_ON_XK: Writes to ahbslv will only be issued to the MC when all prior writes in different XKB pages have reached the point of coherency. This is needed because the MC allows re-ordering of transactions. If this is not set, an entity which polls memory for a descriptor or flag to indicate some other memory has been written, may think something is available in memory when it is not. Note that XK = 4K for Tegra X1 devices. 0 = DISABLE 1 = ENABLE
7	0x1	NONE	DONT_SPLIT_AHB_WR: AHB slave gizmo (memory controller) – Do not split AHB write transaction 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	NONE	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo (memory controller) - Accept AHB write request always. 1= Always accept AHB write request without checking whether there is room in the queue to store the write data. Bypass Memory Controller AHB slave gizmo write queue. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. Memory controller AHB slave gizmos write queue is used in this case. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENABLE	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	NONE	FORCE_TO_AHB_SINGLE: AHB slave gizmo (memory controller) - Force all AHB transaction to single data request transaction 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	NONE	ENABLE_SPLIT: AHB slave gizmo (memory controller) - Enable splitting AHB transaction. 0 = DISABLE 1 = ENABLE

19.3.1.2 AHB_GIZMO_APB_DMA_0

AHB Gizmo APB-DMA Control Register

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.

Bit	Reset	Description
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

19.3.1.3 AHB_MASTER_SWID_0

Refer to Section 18.5.3: Software Client-Groupings in the Memory Controller chapter for usage details.

AHB Master SWID[0] Register

Writes are secure.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx0x000xxxxxxx0000x00)

Bit	Reset	Description
18	0x0	USB2:SWID[0] for USB2
14	0x0	SE:SWID[0] for SE
6	0x0	USB1:SWID[0] for USB1
4	0x0	ARC: SWID[0] for ARC
1	0x0	COP:SWID[0] for COP
0	0x0	CPU:SWID[0] for CPU

19.3.1.4 AHB_GIZMO_USB_0

AHB Gizmo USB Control Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

19.3.1.5 AHB_GIZMO_AHB_XBAR_BRIDGE_0

AHB Gizmo AHB XBAR Bridge Control Register

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x0000008d (0bxxxxxxxxxxxxxxxxxxxxxxxxx10001101)

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
5:4	0x0	MAX_IP_BURSTSIZE: AHB slave gizmo - Maximum allowed IP burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

Bit	Reset	Description
3	0x1	IMMEDIATE: AHB slave gizmo - Start write request to device immediately. 1 = Start write request on the device side as soon as the AHB master puts data into the gizmos queue. 0 = Start the device write request only when the AHB master has placed all write data into the gizmos queue. 0 = DISABLE 1 = ENABLE
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

19.3.1.6 AHB_GIZMO_CPU_AHB_BRIDGE_0

AHB Gizmo CPU AHB Bridge Control Register

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxxx110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

19.3.1.7 AHB_GIZMO_COP_AHB_BRIDGE_0

AHB Gizmo COP AHB Bridge Control Register

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxxx110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master

Bit	Reset	Description
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

19.3.1.8 AHB_GIZMO_XBAR_APB_CTLR_0

AHB Gizmo XBAR APB Control Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx001xxx)

Bit	Reset	Description
5:4	0x0	MAX_IP_BURSTSIZE: AHB slave gizmo - Maximum allowed IP burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
3	0x1	IMMEDIATE: AHB slave gizmo - Start write request to device immediately. 1 = Start write request on the device side as soon as the AHB master puts data into the gizmos queue. 0 = Start the device write request only when the AHB master has placed all write data into the gizmos queue. 0 = DISABLE 1 = ENABLE

19.3.1.9 AHB_MASTER_SWID_1

Refer to Section 18.5.3: Software Client-Groupings in the Memory Controller chapter for usage details.

AHB Master SWID[1] Register

Writes are secure.

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx0x000xxxxxxx0000x00)

Bit	Reset	Description
18	0x0	USB2:SWID[1] for USB2
14	0x0	SE:SWID[1] for SE
6	0x0	USB1:SWID[1] for USB1
4	0x0	ARC: SWID[1] for ARC
1	0x0	COP:SWID[1] for COP
0	0x0	CPU:SWID[1] for CPU

19.3.1.10 AHB_GIZMO_SE_0

AHB Gizmo SE Control Register

Offset: 0x50 | Read/Write: R/W | Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxx) | Default: 0x00000000

Bit	Reset	SW Default	Description
31:24	0x0	NONE	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	DISABLE	DISABLE	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	DMA_BURST_8WORDS	NONE	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

19.3.1.11 AHB_GIZMO_TZRAM_0

AHB Gizmo AHB TZRAM Control Register

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000081 (0bxxxxxxxxxxxxxxxxxxxxxxxxx10xxx001)

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

19.3.1.12 AHB_GIZMO_USB2_0

USB2 master gizmo configuration.

AHB Gizmo USB2 Control Register

Offset: 0x7c | Read/Write: R/W | Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

19.3.1.13 AHB_GIZMO_ARC_0

AHB Gizmo ARC Control Register

MC Master Gizmo Configuration

Offset: 0x98 | Read/Write: R/W | Reset: 0x00060000 (0b00000000xxxxx110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	DISABLE	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	DMA_BURST_8WORDS	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

19.3.1.14 AHB_AHB_WRQ_EMPTY_0

Offset: 0xc4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	COP_AHB_WRQ_EMPTY
0	X	CPU_AHB_WRQ_EMPTY

19.4 AHB Memory Controller Slave

The AHB memory controller slave is the path used by AHB bus masters to access the Memory Controller. It provides access to DRAM from the AHB bus and can pre-fetch data.

19.4.1 AHB Memory Pre-Fetcher

The AHB memory controller slave contains a pre-fetcher block. This is intended to improve performance for AHB masters performing sequential reads from DRAM. Performance can be a problem because the AHB bus protocol only allows a single outstanding read request from a master, so the round trip latency limits bandwidth. As there are two level of arbitration, firstly on AHB and secondly within the memory controller, this latency can become large on a busy system.

There are eight such pre-fetchers, allowing this function to be active for up to eight AHB masters.

When the pre-fetcher is enabled, the first read request initiates a speculative read of up to 128 bytes, and subsequent linear reads will return the data directly to the AHB master without going through the memory controller to DRAM.

19.4.1.1 Pre-Fetch Invalidate

The pre-fetch buffer's contents are invalidated under any of the following conditions, in all cases a read refers to a DRAM read by the assigned AHB master:

- The programmed time-out value is reached since the last read
- A read occurs which is not sequential with the previous read
- A read occurs which is not the same size as the previous read, unless the corresponding DISABLE_CHECK_SIZE_MASTERx bit is set.

- A read occurs which is past the end of the pre-fetch buffer (this will trigger a new fill of the buffer).
- The pre-fetcher is disabled. A momentary disable and then re-enable can be used to invalidate the buffer.

19.4.1.2 Pre-Fetch Buffer Coherency

As the pre-fetch buffer contains a copy of data in DRAM, there is an inherent coherency risk if the DRAM data is updated. The hardware invalidation mechanisms provide some protection here, but there remains risk of subtle problems arising from this hardware. If an AHB master shows errors that appear to arise from stale data then a reasonable experiment is to disable the pre-fetcher to see if that cures the problem. If so, the following guidelines may help.

Problems have been seen for control structures such as USB Transfer Descriptors.

Two approaches can resolve this problem:

1. Pad the data

   As the pre-fetcher invalidates the buffer for any non-sequential read, placing some padding will prevent pre-fetch issues. A problem can be triggered by using something like this:

   ```
   struct dtd { unsigned HW_descriptor[K] ; } ; struct dtd dtd_array[N] ;
   ```

   This makes the hardware descriptors structures as read by the USB DMA sequential, and so pre-fetchable.
   If the definition is modified as:

   ```
   struct dtd { unsigned HW_descriptor[K] ; unsigned Padding ; } ; struct dtd dtd_array[N] ;
   ```

   Then reads by USB DMA of consecutive hardware descriptors become **not** sequential and so there is no coherency issue as the pre-fetch buffer will be invalidated.

2. Invalidate the pre-fetch buffer

   After updating a memory structure, the pre-fetch buffer can be invalidated by disabling the pre-fetcher and then immediately re-enabling it.

19.4.2 AHB Memory Controller Slave Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

19.4.2.1 AHB_AHB_MEM_PREFETCH_CFG5_0

0x6000_C0CC: ahbslv prefetch cfg5 --> reset value = 0x1883.0800.

Offset: 0xcc │ Read/Write: R/W │ Reset: 0x14800800 (0b00010100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable

Bit	Reset	Description
30:26	0x5	AHB_MST_ID: 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0D 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

19.4.2.2 AHB_AHB_MEM_PREFETCH_CFG6_0

0x6000_C0D0: ahbslv prefetch cfg6 --> reset value = 0x1883.0800.

Offset: 0xd0 │ Read/Write: R/W │ Reset: 0x18800800 (0b00011000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable

Bit	Reset	Description
30:26	0x6	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0D 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

19.4.2.3 AHB_AHB_MEM_PREFETCH_CFG7_0

0x6000_C0D4: ahbslv prefetch cfg7 --> reset value = 0x1883.0800.

Offset: 0xd4 │ Read/Write: R/W │ Reset: 0x1c800800 (0b00011100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable

Bit	Reset	Description
30:26	0x7	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0D 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

19.4.2.4 AHB_AHB_MEM_PREFETCH_CFG8_0

0x6000_C0D8: ahbslv prefetch cfg8 --> reset value = 0x1883.0800.

Offset: 0xd8 │ Read/Write: R/W │ Reset: 0x20800800 (0b00100000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable

Bit	Reset	Description
30:26	0x8	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0D 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

19.4.2.5 AHB_AHB_MEM_PREFETCH_CFG_X_0

If DISABLE_CHECK_SIZE is 0, then only read requests that have the exact same size as the original read request that kick-started the prefetch process will cause a "hit". In addition, the address must be the exact next one in the sequence.

For instance, if the first request on a prefetch-enabled AHB Master arrives with SIZE=ONE_BYTE and ADDR[31:0]=0x3, then the next access must have SIZE=ONE_BYTE and ADDR[31:0]=0x4 in order to be considered a hit.

If DISABLE_CHECK_SIZE is 1, then a read request will hit as long as the incoming ADDR[31:4]matches the expected_ADDR[31:4]. expected_ADDR[31:4] is always either "last ADDR[31:4]" or "last ADDR[31:4] + 1", where last ADDR is the prior read request actually issued by the AHB Master(as opposed to the last request to the CIF, which could have been a speculative read).

expected_ADDR[31:4] is always "last ADDR[31:4]" unless "SIZE" field in the last request uses the last byte in the 16-byte CIF word.

Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	DISABLE_ADDR_BNDY_CHK_MST8:
14	0x0	DISABLE_ADDR_BNDY_CHK_MST7:
13	0x0	DISABLE_ADDR_BNDY_CHK_MST6:
12	0x0	DISABLE_ADDR_BNDY_CHK_MST5:
11	0x0	DISABLE_ADDR_BNDY_CHK_MST4:
10	0x0	DISABLE_ADDR_BNDY_CHK_MST3:
9	0x0	DISABLE_ADDR_BNDY_CHK_MST2:
8	0x0	DISABLE_ADDR_BNDY_CHK_MST1

Bit	Reset	Description
7	0x0	DISABLE_CHECK_SIZE_MASTER8:
6	0x0	DISABLE_CHECK_SIZE_MASTER7:
5	0x0	DISABLE_CHECK_SIZE_MASTER6:
4	0x0	DISABLE_CHECK_SIZE_MASTER5:
3	0x0	DISABLE_CHECK_SIZE_MASTER4:
2	0x0	DISABLE_CHECK_SIZE_MASTER3:
1	0x0	DISABLE_CHECK_SIZE_MASTER2:
0	0x0	DISABLE_CHECK_SIZE_MASTER1:

19.4.2.6 AHB_ARBITRATION_XBAR_CTRL_0

XBAR Control Register

Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxx00)

Bit	Reset	Description
16	0x0	MEM_INIT_DONE: Software should set this bit when memory has been initialized 0 = NOT_DONE 1 = DONE
1	0x0	HOLD_DIS: By default CPU accesses to IRAMs will be held if there are any pending requests from the AHB to the IRAMs. This is done to avoid data coherency issues. If software handles coherency then this can be turned off to improve performance. Software writes to modify. 0 = ENABLE 1 = DISABLE
0	0x0	POST_DIS: Software writes to modify 0 = ENABLE 1 = DISABLE

19.4.2.7 AHB_AHB_MEM_PREFETCH_CFG3_0

See the description of the pre-fetcher above. 6000:C0E4: ahbslv prefetch cfg3 --> reset value = 0x1483.0800.

Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x0c800800 (0b00001100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable. 0=disable

Bit	Reset	Description
30:26	0x3	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = UNUSED_09 10 = UNUSED_0A 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0D 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNISED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

19.4.2.8 AHB_AHB_MEM_PREFETCH_CFG4_0

See the description of the pre-fetcher above. 6000:C0E8: ahbslv prefetch cfg4 --> reset value = 0x1483.0800.

Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x10800800 (0b00010000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable. 0=disable

Bit	Reset	Description
30:26	0x4	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = UNUSED_09 10 = UNUSED_0A 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0B 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

19.4.2.9 AHB_AVP_PPCS_RD_COH_STATUS_0

0x6000_C0EC: ARM7 outstanding rd/wr

Offset: 0xec │ Read/Write: RO │ Reset: 0x000X000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	RDS_OUTSTANDING
0	X	WRS_OUTSTANDING

19.4.2.10 AHB_AHB_MEM_PREFETCH_CFG1_0

NV_ahbslvmem prefetch

The NV_ahbslvmem prefetch feature reduces latency and improves overall bandwidth for AHB Masters doing reads to SDRAM. AHB only allows one outstanding read transaction at a time. When enabled, and kick-started by a first read request (which will "miss" since there would be nothing in the prefetch FIFO) the prefetcher makes speculative read requests to the memory controller in consecutive progression of linear address. Without this feature, an AHB Master will suffer roundtrip latency through the gizmo, PPCS, CIF, MC arbitration, and all the read data passing for every AHB transaction.

Address Boundaries for prefetching will stop the sequential prefetch process from making additional speculations. This is useful to help prevent coherency issues -- software needs ways to stop the prefetcher from prefetching data before the data has been written in memory. If an AHB master can be known to make 256 byte transfers that are aligned accesses, ADDR_BNDRY should be set to a value of 4.

INACTIVITY_TIMEOUT is intended to prevent coherency problems. If no read request from the prefetch-enabled master is observed after the number of cycles specified in the inactivity timeout counter, then any speculatively prefetched read data is invalidated.

There are eight AHB masters that can be enabled for prefetching. Each prefetch buffer can hold up to 8 entries (of 128 bits each entry) of prefetched data.

0x6000_C0F0: ahbslv prefetch cfg1 --> reset value = 0x1483.0800.

Offset: 0xf0 | Read/Write: R/W | Reset: 0x04800800 (0b00000100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x1	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = UNUSED_09 10 = UNUSED_0A 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0D 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

19.4.2.11 AHB_AHB_MEM_PREFETCH_CFG2_0

See the description of the pre-fetcher above. 0x6000_C0F4: ahbslv prefetch cfg2 --> reset value = 0x1883.0800.

Offset: 0xf4 | Read/Write: R/W | Reset: 0x08800800 (0b00001000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable

Bit	Reset	Description
30:26	0x2	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = UNUSED_09 10 = UNUSED_0A 11 = UNUSED_0B 12 = HSMMC 13 = UNUSED_0D 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = UNUSED_13 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT

19.4.2.12 AHB_AHBSLVMEM_STATUS_0

0x6000_C0F8: ahbslv outstanding rd, rdque_empty status

Offset: 0xf8 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	PPCS_RDS_OUTSTANDING
0	X	GIZMO_IP_RDQUE_EMPTY

19.4.2.13 AHB_ARBITRATION_AHB_MEM_WRQUE_MST_ID_0

0x6000_C0FC: AHB Memory Write Queue AHB Master ID Register

Offset: 0xfc │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	AHB_MASTER_ID: 0 = There is no write data in the write queue from that AHB master.

19.4.2.14 AHB_ARBITRATION_CPU_ABORT_ADDR_0

0x6000_C100: CPU Abort Address Register

Offset: 0x100 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort.

19.4.2.15 AHB_ARBITRATION_CPU_ABORT_INFO_0

0x6000_C104: CPU Abort Info Register

Offset: 0x104 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN
12	X	IRAMD: Abort occurred due to an iRAMd protection violation 0 = ABT_DIS 1 = ABT_EN
11	X	INV_IRAM: Abort occurred due to an access to invalid iRAM address space 0 = ABT_DIS 1 = ABT_EN
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN
7	X	CACHE: Abort occurred due to a Cache protection violation 0 = ABT_DIS 1 = ABT_EN
6	X	PROTECTION: TRUE for any protection violation 0 = ABT_DIS 1 = ABT_EN
5	X	ALIGN: TRUE for abort caused by Misalignment (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
4	X	BADSIZE: TRUE for abort caused by Bad Size (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
3	X	WRITE: Aborted transaction was a Write 0 = ABT_DIS 1 = ABT_EN
2	X	DATA: Aborted transaction was a Data access 0 = ABT_DIS 1 = ABT_EN
1:0	X	SIZE: Aborted transaction Request Size 00=byte 01=halfword 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

19.4.2.16 AHB_ARBITRATION_COP_ABORT_ADDR_0

0x6000_C108: CPU Abort Address Register

Offset: 0x108 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort

19.4.2.17 AHB_ARBITRATION_COP_ABORT_INFO_0

0x6000_C10C: COP Abort Info Register

Offset: 0x10c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN
7	X	CACHE: Abort occurred due to a Cache protection violation 0 = ABT_DIS 1 = ABT_EN
6	X	PROTECTION: TRUE for any protection violation 0 = ABT_DIS 1 = ABT_EN
5	X	ALIGN: TRUE for abort caused by Misalignment (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
4	X	BADSIZE: TRUE for abort caused by Bad Size (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
3	X	WRITE: Aborted transaction was a Write 0 = ABT_DIS 1 = ABT_EN
2	X	DATA: Aborted transaction was a Data access 0 = ABT_DIS 1 = ABT_EN

Bit	Reset	Description
1:0	X	SIZE: Aborted transaction Request Size 00=byte 01=halfword 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

19.4.2.18 AHB_AHB_SPARE_REG_0

0x6000_C110: AHB Spare Register Bits

Offset: 0x110 | Read/Write: R/W | Reset: 0x00000060 (0b00000000000000000000xxxxx1100000)

Bit	Reset	Description
31:12	0x0	AHB_SPARE_REG: AHB Spare Register
4:0	0x0	CSITE_PADMACRO_TRIM_SEL: Trimmer select register for CoreSight clock pad macro.

19.4.2.19 AHB_XBAR_SPARE_REG_0

0x6000_C114: XBAR Spare Register Bits

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:12	0x0	UNUSED: XBAR Diagnostic Register Spare Bits
11	0x0	DISABLE_XBAR_ABORT_EXTENSION: XBAR Diagnostic Register to disable one clock extension of u_abort
10	0x0	DISABLE_COP_BYTE_WR: XBAR Diagnostic Register to disable COP byte writes
9	0x0	DISABLE_XUSB_DEV_BYTE_WR: XBAR Diagnostic Register to disable XUSB dev byte writes
8	0x0	DISABLE_XUSB_HOST_BYTE_WR: XBAR Diagnostic Register to disable XUSB host byte writes
7	0x0	DISABLE_SDMMC_BYTE_WR: XBAR Diagnostic Register to disable SDMMC byte writes
6	0x0	DISABLE_XBAR_APB_BYTE_WR: XBAR Diagnostic Register to disable byte writes to APB slaves
5	0x0	MASK_PEND_IRAM_REQ: XBAR Diagnostic Register to mask pending IRAM request while arbitration
4	0x0	KILL_NEXT_REQ_ON_ABORT_UCQ: XBAR Diagnostic Register to kill next request on ABORT from UCQ
3	0x0	KILL_NEXT_REQ_ON_ABORT_AHB: XBAR Diagnostic Register to kill next request on ABORT from AHB Bridge
2	0x0	KILL_NEXT_REQ_ON_ABORT_VCP2: XBAR Diagnostic Register to kill next request on ABORT from VCP2
1	0x0	KILL_NEXT_REQ_ON_ABORT_COP: XBAR Diagnostic Register to kill next request on ABORT from COP
0	0x0	KILL_NEXT_REQ_ON_ABORT_APC: XBAR Diagnostic Register to kill next request on ABORT from APC

19.4.2.20 AHB_AVPC_MCCIF_FIFOCTRL_0

Note: *The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.*

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC sides of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC

A '1' written to the CCLK override field keeps the client's clock always on inside the MCCIF.

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Offset: 0x120 | Byte Offset: 0x480 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	AVPC_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	AVPC_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	AVPC_CCLK_OVERRIDE
17	0x0	AVPC_RCLK_OVERRIDE
16	0x0	AVPC_WCLK_OVERRIDE
3	DISABLE	AVPC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	AVPC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	AVPC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	AVPC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

19.4.2.21 AHB_TIMEOUT_WCOAL_AVPC_0

Note: *Write coalescing is no longer supported by the MCCIF clients. Registers are retained for software compatibility but are not used by the hardware.*

Write Coalescing Time-Out Register

Offset: 0x124 | Byte Offset: 0x490 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	AVPCARM7W_WCOAL_TMVAL

19.4.2.22 AHB_MPCORE_MCCIF_FIFOCTRL_0

Note: *The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility*

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC sides of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation, where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Offset: 0x128 | Byte Offset: 0x4a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	LEGACY	SYS_REGS_MPCORE_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	SYS_REGS_MPCORE_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	SYS_REGS_MPCORE_CCLK_OVERRIDE
17	0x0	SYS_REGS_MPCORE_RCLK_OVERRIDE
16	0x0	SYS_REGS_MPCORE_WCLK_OVERRIDE

CHAPTER 20: PIXEL MEMORY FORMAT

This document outlines the memory tiling formats that may be produced and consumed by interacting engines in Tegra® X1. This document also provides a description of the pixel formats that may be produced and consumed by interacting engines in Tegra X1.

20.1 Tiling Formats

Tegra X1 supports the memory tiling formats described below. Clients can support block linear formats and/or pitch linear formats. Refer to the description of each device for more details on the modes supported. Pitch linear client support is useful for clients that interact directly or indirectly with the CPU or that make horizontal (or linear) accesses to memory. Block linear is optimized for clients that access regions of pixels across more than one line.

Table 83: Client Shared Tiling Formats

Format	Status	Used for	Description
Pitch Linear	Supported by some clients	Software-compatibility, clients that linearly access memory	Simple array ordering
Block linear 16Bx2 with Tegra sector ordering	Supported by most clients	GPU color surfaces, clients that access memory in blocks or vertically	Square DRAM page arrangement (blocks) with folded atoms

The tiling format describes the organization of bytes within a surface. The tiling formats convert an (x, y) offset of a sample in a surface to a linear address. To do the translation, they need additional information about the surface.

Table 84: Surface Descriptor Parameters

Parameter	Used by	Description	Values	Minimum Alignment[a]
Layout	All	Surface layout	PITCH, BLOCKLINEAR	N/A
Base Address	Pitch	Starting address of surface, in bytes	32-bit virtual address (40-bit virtual address within the GPU) -or- 34-bit physical address	64 B (general) 128 B (GPU) 256 B (NVENC, VIC, Display)
Pitch	Pitch	Line-to-line stride in bytes	Large enough for at least 4,096 pixels in bytes (some clients may support larger values)	64 B (general) 128 B (GPU) 256 B (NVENC) up to 256 B (VIC)[b]
Base Address	Block Linear	Starting address of surface, in bytes	32-bit virtual address (40-bit virtual address within the GPU)-or- 34-bit physical address	512 B (one gob)
Block Width	Block Linear	Width of block in gobs	ONE_GOB (this is the only setting supported on Tegra X1)	N/A
Block Height	Block Linear	Height of block in gobs	[ONE_GOB, THIRTYTWO_GOBS] (SIXTEEN_GOBS is the normal setting, but TWO_GOBS may be needed for some display surfaces with rotation disabled.	N/A
Block Depth	Block Linear	Depth of block in gobs	[ONE_GOB, THIRTYTWO_GOBS] (values greater than ONE_GOB are currently only supported by the GPU)	N/A
Width	Block Linear	Width of surface in samples	[1, 4096] samples (some clients may support larger values)	1 sample

a. These are minimum alignments. Larger alignments are sometimes needed for performance

b. The pitch alignment restrictions for VIC depend on its cache line shape: 256Bx1: 256B, 128Bx2: 128B, otherwise: 64B

The Maxwell 3d class supported by the GPU has a full list of surface parameter ranges that, in many cases, exceed those in this table. For GPU surfaces that are larger than a Tegra client can support, all manipulation of the surface must be done on the GPU. Displayable surfaces are limited to the max 8K image width supported by display.

20.1.1 DRAM Organization

Although the tiling equations produce byte addresses, the memory controller interfaces operate in terms of *atoms*, which are the smallest possible unit of addressable memory. Legacy chips have used a 16-byte atom. Tegra X1 uses a 64-byte atom.[1]

Tegra X1 works with an effective 1 KB or 2 KB page, *i.e.* bank bits change on 1 KB or 2 KB boundaries. DRAM accesses are most efficient if adjacent accesses remain in the same page. Accesses can cross a page boundary into another bank as long as there are sufficient accesses in the current page to allow the DRAM controller to overlap the opening of the new bank while the current one is accessed. The *row cycle time* is the minimum amount of time that it takes the DRAM to open and close a page; in order to efficiently access the DRAM the client must arrange to issue enough requests within each bank to cover the row cycle time across all available banks.

If a client accesses different pages in the same bank, this produces a *bank conflict*. Bank conflicts are very inefficient, since they cause the DRAM to close and re-open the page. Unless the controller can find enough work in other banks to cover this cost, the bank conflict will turn into lost efficiency. The purpose of most of the tiling formats is to maximize the number of requests in one page and remove or reduce the potential for bank conflicts.

In physically allocated memory (carveout), bank swizzling equations ensure that banks are evenly interleaved, so bank conflicts are rare. In dynamically allocated virtual memory, a given 4 KB page only contains data from 4 banks. To prevent unnecessary bank conflicts, memory allocation should use page coloring. In page coloring, each 4 KB physical page is assigned a “color” based on the banks it contains. For example, a “red” page might have banks 0-3 and a “green” page might have banks 4-7. The page table mapping should map physical pages of the appropriate color to virtual pages so as to optimally interleave the banks. If page coloring is not feasible, clients should take advantage of the fact that there are no bank conflicts within a single 4 KB virtual memory page, but need to be aware that bank conflicts can occur at the boundary of this page with other virtual memory pages.

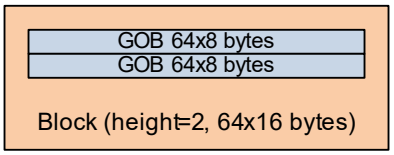
20.1.2 Block Linear

The block linear formats map batches of contiguous addresses into rectangular *blocks*, and tile these blocks linearly. Blocking serves two purposes:

1. It maps a DRAM page into a rectangular region, rather than into a one-pixel-tall strip, to capture locality in a graphics geometry stream or other 2d-oriented client, such as an MPEG encoder or decoder
2. It makes it possible to predictably interleave banks in x and y, minimizing bank conflicts over a large region.

Blocks themselves are arranged from *GOB*s, which are *Groups of Bytes*. A Maxwell GOB, as used in Tegra X1, is 512 bytes arranged as 64x8 bytes. GOBs are stacked vertically to form a block. The number of GOBs stacked vertically in a block is controlled by an additional surface parameter called the *block height*. The recommended block height for most buffers on Tegra X1 is 16 GOBs (128 lines). This supports 8x8 bank interleaving. For buffers for which linear display access is more important than access by GPU, VIC, or other block-oriented engine, block height may be set to 1 GOB, providing more locality for the display client.

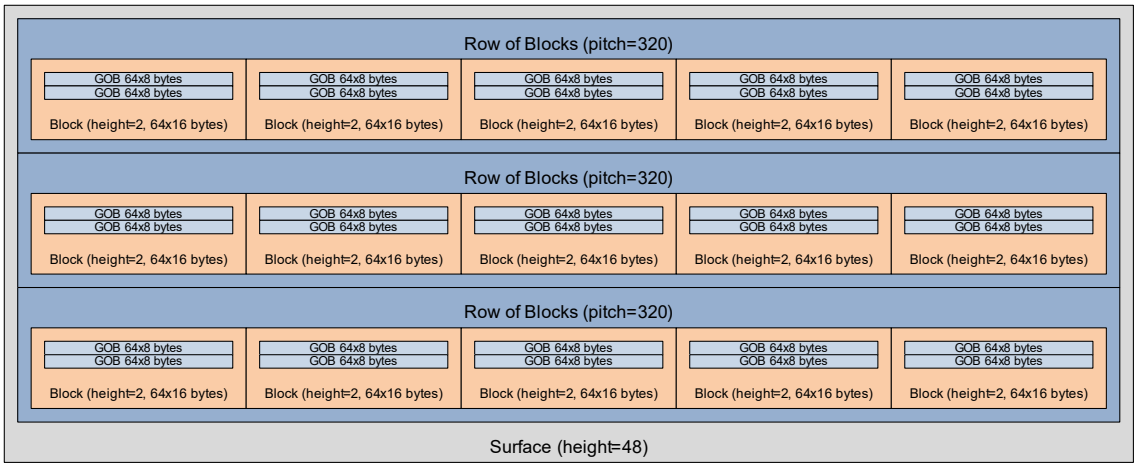
Figure 45: Example block with block_height = 2



1. Certain clients can take advantage of the memory system’s sub-partition feature to read or write non-contiguous pairs of 32B subpackets, rather than being restricted to a 64B DRAM atom. See the following section for details.

Blocks are arranged horizontally into a *Row of Blocks (ROBs)* to fill out the surface to the width value (or slightly beyond, if the width is not a multiple of the 64 B block width). Upper address bit swizzling in the bank swizzling algorithm generally makes it unnecessary to pad the surface width further.

Figure 46: Block Linear Surfaces



The tiling equations for block linear start out by computing the *GOB address* as follows:

```
GOB_address = base_address +
(y / (8 * block_height)) * 512 * block_height * image_width_in_gobs +
(x * bytes_per_pixel / 64) * 512 * block_height +
(y % (8 * block_height) / 8) * 512
where
image_width_in_gobs = ceiling(image_width * bytes_per_pixel / 64)
```

The individual pixel byte address is then determined from the GOB address based on the sector packing and sector ordering.

Note: *Image_width_in_gobs is directly determined by the image width itself. There is no additional 'pitch' parameter, as in legacy Tegra tiling. Tegra X1 units must not make data fetches wider than 64B (the block width) to block linear surfaces.*

20.1.2.1 Sector Packing and Sector Ordering

Sector packing is the arrangement of bytes within a 32 B sector. Sector ordering is the arrangement of sectors within a GOB. All buffers that are shared between Tegra clients use 16 B x 2 sector packing: the 32 B within a sector are arranged as two 16 B lines on top of each other.

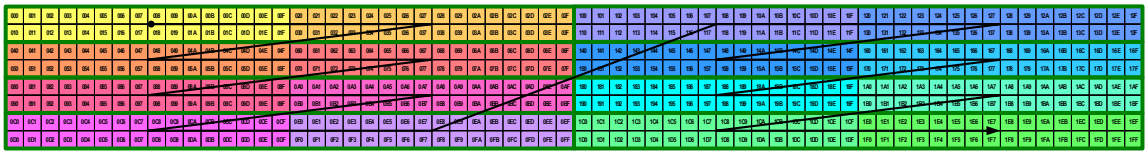
Note: *The GPU supports KINDs with other packing, but these are private to the GPU.*

The sectors are arranged in the GOB as shown below. This format will be the lingua franca between Tegra all engines, including the Maxwell GPU.

The diagram below represents a view of a GOB in pixel space. Each numbered box represents a byte, and each number represents the address offset of that byte from the base address of the GOB. The green outline denotes the set of sectors that make up a 64 B memory atom.

As an example of how to interpret the diagram below, assuming 4 bytes per pixel, the first 8 pixels in row 0 within the GOB are stored in bytes 00-0F and 20-2F (i.e., first, second, third, fourth pixels are stored in bytes 00-03, 04-07, 08-0B, 0C-0F respectively, and fifth, sixth, seventh, eighth pixels are stored in bytes 20-23, 24-27, 28-2B, 2C-2F respectively) in one 64 byte atom. The ninth pixel in row 0 within the GOB is stored in bytes 100-103 in a separate 64 byte atom.

Figure 47: Block Linear (GPU 16Bx2 kind with Tegra X1 Sector Ordering)



The address of a byte is computed as follows:

Address = GOB_addr + ((x%64)/32)*256 + ((y%8)/2)*64 + ((x%32)/16)*32 + (y%2)*16 + (x%16)

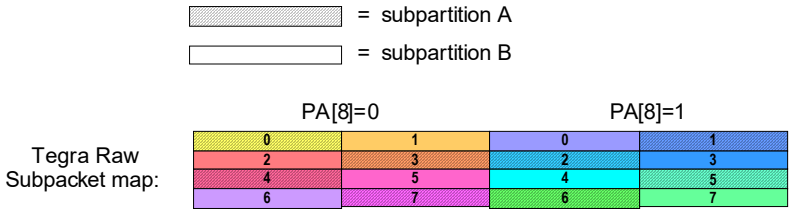
The sector ordering in this format reduces the amount of buffering required by clients that use line stores, such as Display and ISP. It also reduces the amount of instantaneous bandwidth required by Display because the number of lines that must be fetched before scanning out is reduced. Clients that do more vertically oriented accesses can take advantage of the sub-partition feature described below.

20.1.3 Sub-Partitions

The 64 b DRAM channel can be divided into two sub-partitions with DQ[31:0] forming sub-partition A and DQ[63:32] forming sub-partition B.

Each 64 B atom is divided into 2 sectors of 32 B. One sector contains bytes 0 – 31 while the other contains bytes 32 – 63. The sectors are mapped to sub-partitions by the MCCIF using PA[8]^PA[6]^PA[5]. This produces a checkerboard mapping of the physical address space to sub-partitions which allows both sub-partitions to be used whether the primitive is oriented horizontally or vertically. The checkerboard is mirrored between tiles in a GOB which is important when the surface is compressed.

Figure 48: Address to Sub-Partition Mapping



Clients that use the sub-partition feature can access two non-contiguous sectors targeting the same GOB in a single request provided they do not target the same sub-partition. A common use for this feature is to change the shape of a 64 B request from 32 B x 2 (without sub-partitions) to 16 B x 4 (using sub-partitions), which is the preferred fetch footprint for the GPU texture cache and is used by display when scanning out vertically.

20.2 Hardware Support

Clients implement registers in their module to control the four parameters (base address, kind, width or pitch, and block height) required for tiling control of each surface supported by the client.

For planar and semi-planar surfaces, in which a single image is composed of multiple buffers or *planes*, independent parameters should be provided for each buffer or *plane*.[1]

The kind and block height fields are enumerations defined below. All Tegra X1 clients use the same enumeration values. The most common block height for Tegra X1 is expected to be SIXTEEN_GOBS, but the others are supported.

Table 85: Block Height Enumeration

Block height enumeration	Value	Encoding (four bits log2 encoding)
ONE_GOB	1	0

1. Multiple YUV surface producers and consumers violate this rule and use one kind and block_height parameter for all planes in a YUV image, which is problematic since the planes have different numbers of pixels.

Table 85: Block Height Enumeration

Block height enumeration	Value	Encoding (four bits log2 encoding)
TWO_GOBS	2	1
FOUR_GOBS	4	2
EIGHT_GOBS	8	3
SIXTEEN_GOBS	16	4
THIRTYTWO_GOBS	32	5

Surface kinds shared by Tegra X1 clients are listed below. Tegra X1 clients expecting to share surfaces must support the NV_MMU_PTE_KIND_GENERIC_16Bx2 surface kind listed below, and optionally support the NV_MMU_PTE_KIND_PITCH surface kind. Note that the NV_MMU_PTE_KIND_GENERIC_16Bx2 enumeration is used by Tegra clients to represent the layout described in Tegra systems, and the same enumeration is used by the discrete GPU group to represent a different memory layout in non-Tegra systems.

Table 86: Surface Kind Enumeration

Surface kind enumeration	Encoding (eight bits)
NV_MMU_PTE_KIND_PITCH	0x00
NV_MMU_PTE_KIND_GENERIC_16Bx2	0xFE

Tegra X1 clients should make memory fetches that are integral multiples of DRAM atoms to avoid wasting bandwidth. If fetches can extend beyond the active buffer size (in width or height), the buffer must be padded to accommodate the maximum sized fetch. For pitch linear surfaces, this may mean padding the pitch and/or padding the number of lines in the buffer. For block linear surfaces (in rare instances), this may mean adding an additional row of blocks at the bottom of the buffer.

Note: *For block linear surfaces, there is no pitch parameter separate from surface width. To avoid making data fetches outside a buffer's allocated width, units should generally not make requests that are wider than 64B (the block width).*

20.3 Software Guidelines

Software should allocate surfaces respecting the minimum alignments. This section gives additional alignment restrictions for certain clients and advice on allocating surfaces for best performance.

Certain clients, such as NVENC, NVDEC, Display and VIC use 32-bit registers (left-shifted 8) to specify 40-bit base addresses of a buffer. Base addresses for these clients must be aligned to a minimum of 256 B.

Certain clients, such as NVENC and VIC have a fixed cache line size and read all data within a cacheline, even if it is beyond the nominal height and width of the surface. For such clients, surface width and height need to be padded to at least the next multiple of the cache line size so the unit does not read beyond the end of the buffer.

Surface type

- **Block linear** format generally provides best performance and should be chosen by default.
- **Pitch format** may be preferable if producer and consumer clients perform linear accesses and both clients support pitch.
- **Private formats** are used by a few units, such as NVENC and NVDEC. For private formats, the client-required allocation and padding rules must be followed.

Block Linear

- **Alignment**. Align surface start address as follows:

- 512 B is the minimum alignment for correct behavior. Performance will be poor (no zero-bandwidth clears, poor page locality).
- 1 KB alignment is better. Zero bandwidth clears will work. Bank interleave will be suboptimal.
- 8 KB alignment is best. All compression features work. Bank interleave assumes 8 KB alignment. Use this setting for large buffers.

- **Block height** should normally be programmed to SIXTEEN_GOBS (128 lines). For certain surfaces for which display is the primary client, block heights of ONE_GOB or TWO_GOBS are recommended. Surfaces to be accessed by the GPU texture unit may require a smaller block height under certain conditions.[1]
- **Width** is typically prescribed by the application. Hardware allocates a block linear surface to be an integral number of blocks wide, so its allocated width is width*Bpp rounded up to the next block boundary (64B), which is sufficient padding in almost all circumstances.[2]
- **Height padding**. Surfaces should be an integral number of blocks tall. If a client such as NVENC or VIC has a cache line that is taller than the block height, additional row(s) of blocks may be required for padding.

Pitch

- **Alignment**. Align surface start address to a minimum of:
  - 128 B if surface is to be accessed by GPU
  - 256 B if surface is to be accessed by NVENC, NVDEC, Display or VIC (due to the 8-bit left-shift issue)
  - 64 B otherwise
- **Pitch**. Surface pitch must be a multiple of:
  - 128 B bytes if surface is to accessed by GPU
  - 128 B if surface is to be read by VIC with 128x2 cache line shape
  - 256 B if surface is to be read by VIC with 256Bx1 cache line shape
  - 64 B otherwise
- **Height padding**. For clients, such as NVENC and VIC, additional lines of padding may be required beyond the nominal height of the surface. For example, when VIC is reading a source buffer with an internal cache line shape of 32x8, the source buffer must be a multiple of 8 lines tall.

20.3.1 Block Height Limitations

NVENC supports only a subset of the block heights. However, in general all hardware blocks support full range of block heights (i.e. 2/4/8/16/32). Below is a table of programmable block height support for Tegra X1.

Block	Surface	Block Height(s) Supported
ISP	Any	All (optimal performance BlockHeight=2)
NVDEC	Any	All except 1 (optimal performance based on Coding Tree Block size)
NVENC	Input/Reference	All (optimal performance based on Coding Tree Block size)
	Recon	BlockHeight = 2 only
VIC	Input	All (optimal performance BlockHeight=32)
	Output	All (optimal performance BH=32 normal/BH=2 rotated)
Display	Horizontal	All (optimal performance BlockHeight=2)
	Vertical	All (optimal performance BlockHeight=32)

1. For buffers that are to be accessed using the gp10b texture unit, the BlockHeight parameter must match that used by the texture unit. In particular, note that if a texture surface is smaller than the specified size in blocks, texture uses a shrunken BlockHeight instead of the nominal parameter specified in the Texture Header.
2. If VIC cacheline shapes of 128x2 or 256x1 are used (generally not recommended for blocklinear), width must be padded up to a multiple of the cache line width. If this is not possible, a narrower cache line shape must be used.

Beyond limitations of multimedia blocks performance with respect to block height, the SMMU has possibly a larger performance penalty for block heights that are outside of the “SMMU block height sweet spot”.

A summary of block height recommendations is given below:

- BlockHeight = 16 GOBs (128 lines) is the recommended setting for most clients that use block linear. It provides decent accesses per activate for clients that traverse a buffer horizontally, vertically, or in tiles. DRAM pages are 64B x 64 lines. TLB cache lines are roughly square in screen space. *This should be the default setting and is the only performant block height for rotated display.*
- BlockHeight = 1 GOB (8 lines) is optimal for clients that make horizontal accesses, doubling access per activate, but at the expense of clients that make vertical accesses. This is the optimal setting for non-rotated display, but other block heights are okay for non-rotated display.
- BlockHeight = 2 GOBs (16 lines) is the hard-coded value for NVENC video engines RECON surface. It does not provide the access per activate benefit of BlockHeight = 1 and TLB cache lines have a poor aspect ratio. Buffers used by this client must have block height = 2.

20.4 Pixel Formats

20.4.1 Pixel Format Naming

Tegra X1 Mobile pixel format naming conventions continues to use the naming style adopted in previous chips.

20.4.1.1 Naming Conventions

Tegra X1 pixel format names are based on the following set of rules:

3. All format names begin 'T_' (T followed by underscore) to distinguish them from Tegra 3 and other format name conventions.
4. If the number of bits per pixel is not a power of 2, use an Array of Words name.
   Example: T_R8_G8_B8.
   a. Write all the components in increasing address order from left to right with an underscore between each component.
   b. The name lists the bits in this order. Each component is described by a set of attributes:
      - Component letter (see Table 87)
      - a decimal number (the number of bits in the component – always a multiple of 8)
      - an optional Type Suffix (see Table 88)
5. Otherwise, if the number of bits per pixel is a power of 2, use a Single Word name.
   Example: T_A8B8G8R8, T_R5G6B5.
   a. Collect all the components in the pixel into a single word using little endian order.
   b. Arrange the components from left to right in msb to lsb order.
   c. The name lists the bits in this order. Each component is described by:
      - a Component letter (see Table 87).
      - a decimal number (the number of bits in the component)
      - an optional type suffix (see Table 88)
6. Identify the type:
   a. Identify the type of each component, ignoring any S and X components. Applicable types are listed in Table 88.
   b. If all the (non-ignored) types are Unsigned Normalized then do not use any type suffix.
   Example: T_A8B8G8R8.

c. Else if all the (non-ignored) types use the same type (not Unsigned Normalized) then add to the end of the format name an underscore and a type suffix. This suffix applies to all the components (except S and X components). Example: T_A8B8G8R8_I.

d. Else there are different types. Add a type suffix (without an underscore) to each component identifier (following the number of bits). Do not add a type identifier to any S or X or Unsigned Normalized components. Do not add an underscore or type identifier to the end of the format name.
Example: T_A2B10FG10FR10F.

7. Special cases:

a. T_E5B9G9R9_F - each RGB component is a mantissa and E is a shared exponent which is applied to each component. The result is 3 floating point components (RGB).

b. Packed YUV formats like T_V8_Y8__U8_Y8 and T_B8_G8__R8_G8 which have a repeated component: This format represents 2 pixels. The double underscore is the separator between the two pixels. The repeated component (Y/G) is unique to each pixel. The other components (U/V/R/B) are shared by both pixels. Use Array of Words naming for these (see Step 2 above).

c. Bayer formats (TBD).

d. Compressed formats (ETC, DXT, LATC, EAC).

e. Append the “N” delimitor to indicate chroma subsampling (i.e., N444, N422, N422R, N420). For example, T_U8V8_N420 indicates a U8V8 plane with 4:2:0 chroma subsampling).

8. Multi-Pixel Packing Formats:

If more than one pixel is described by a single word which is a power-of-two bits, then use a word-oriented name.

Example: T_X2Lc10Lb10La10.

a. Collect all the components in the pixel into a single word using little endian order.

b. Arrange the components from left to right in msb to lsb order

c. The name lists the bits in this order. Each component is described by:

- a Component letter (see Table 87)
- a lower case pixel ordering identifier “a”, “b”, “c”, etc, where “a” is the “first” logic pixel in the group, “b” is the second, “c” is the third, and so on. For components that are shared by all pixels in the group, or X components (that are not part of any pixel), the pixel ordering identifier is omitted. For components that are shared by more than one, but not all, pixels in the group, multiple pixel order identifying letters are to be used.
- a decimal number (the number of bits in the component)
- an optional Type Suffix (see Table 88)

9. Multiple Plane Formats:

Some units may use a different color format name for each plane. In that case, the color format names will be as described above (e.g., T_Y8, T_U8, T_V8, T_V8U8, etc). Other units may use a single color format name describing all planes in a single name. In this case, the name has one or more triple underscores. The triple underscore separates one plane from the next.

Example: T_Y8___V8U8_N420:

Indicates a semi-planar YUV with T_Y8 in the first plane, and T_V8U8 in the second plane, with 4:2:0 chroma subsampling.

Example: T_Y8___U8___V8_N420:

Indicates a fully planar YUV with T_Y8 in the first plane, T_U8 in the second plane, and T_V8 in the third plane, with 4:2:0 chroma subsampling.

Note: *Tegra X1 clients are required to support YCbCr Rec.601 formats (unsigned 8 bit values), but for brevity, we represent Cb and Cr components with U and V respectively (e.g., AYUV represents*

a format containing A, Y, Cb, Cr components). This does not imply that "YUV" formats use Y, U, V components (signed 8 bit values with a 128 offset).

Table 87: Pixel Naming Component Letters

Component Letter	Description
R	Red
G	Green
B	Blue
A	Alpha
L	Luma (copy to R, G, and B. Copy from R.)
Y	Luminance for YCbCr601 (YUV)
U	Cb for YCbCr601 (YUV)
V	Cr for YCbCr601 (YUV)
N	Delimiter for indicating chroma subsampling (i.e., _N444, N422, N422R, N420)
P	Palette index
C	Coverage (or other special usage)
S	Stencil (always Unsigned Integer)
Z	Depth value
E	Shared exponent. When this component is present, all the other components are mantissas to which this exponent is applied.
X	Unused (ignored on read)

Table 88: Pixel Naming Type Suffixes

Type Suffix	Description
<none>	Unsigned Normalized
SN	Signed Normalized
I	Signed Integer
UI	Unsigned Integer
NL	Nonlinear Z Format
F	Float
LN	Unsigned Normalized sRGB colorspace (L=log)
MSAA	Multisample (special case – only used with S8 (T_S8_MSAA))
N444	No subsampling of chroma (chroma and luma are full resolution)
N422	Subsampled chroma in 422 (half res horizontal, full res vertical)
N422R	Subsampled chroma in 420R (full res horizontal, half res vertical)
N420	Subsampled chroma in 420 (half res horizontal, half res vertical)
TRUE	Uses true YUV (not YCbCr)

20.4.1.2 Pixel Format Bit Patterns

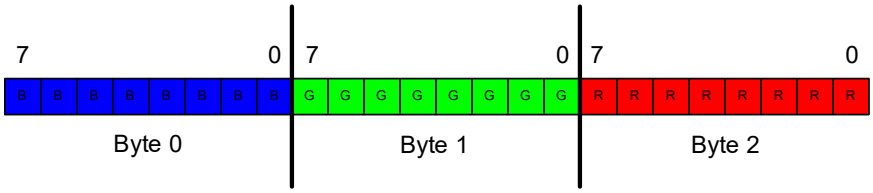
The tables below describe the bit patterns associated with the pixel formats supported in Tegra X1.

The maximum pixel size supported by Tegra X1 engines will be 128 bits.

Array of Words Pixel Formats

Array of Words type pixel formats will list each component in increasing address order from left to right.

e.g., T_B8_G8_R8



e.g., T_R16_G16_B16_I

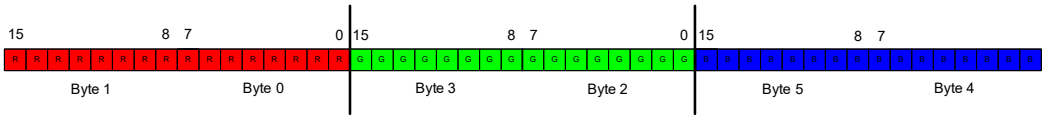


Table 89: Array of Words Pixel Formats

Hardware Name	Bit pattern (Left to Right: LS-Byte to MS-Byte□ms-bit to ls-bit per Byte)	Software Name
T_R8_G8_B8	RRRRRRRR GGGGGGGG BBBBBBBB	NvColorFormat_R8_G8_B8
T_B8_G8_R8	BBBBBBBB GGGGGGGG RRRRRRRR	NvColorFormat_B8_G8_R8
T_R8_G8_B8_SN	RRRRRRRR GGGGGGGG BBBBBBBB	NvColorFormat_SNorm_R8_G8_B8
T_R8_G8_B8_UI	RRRRRRRR GGGGGGGG BBBBBBBB	NvColorFormat_Uint_R8_G8_B8
T_R8_G8_B8_I	RRRRRRRR GGGGGGGG BBBBBBBB	NvColorFormat_Int_R8_G8_B8
T_Y8_U8_V8	YYYYYYYY UUUUUUUU VVVVVVVV	NvColorFormat_Y8_U8_V8
T_Y8_V8_U8	YYYYYYYY VVVVVVVV UUUUUUUU	NvColorFormat_Y8_V8_U8
T_U8_Y8_V8	UUUUUUUU YYYYYYYY VVVVVVVV	NvColorFormat_U8_Y8_V8
T_V8_Y8_U8	VVVVVVVV YYYYYYYY UUUUUUUU	NvColorFormat_V8_Y8_U8
T_U8_V8_Y8	UUUUUUUU VVVVVVVV YYYYYYYY	NvColorFormat_U8_V8_Y8
T_V8_U8_Y8	VVVVVVVV UUUUUUUU YYYYYYYY	NvColorFormat_V8_U8_Y8
T_Y8_U8_V8_TRUE	YYYYYYYY UUUUUUUU VVVVVVVV	NvColorFormat_Y8_U8_V8
T_Y8_V8_U8_TRUE	YYYYYYYY VVVVVVVV UUUUUUUU	NvColorFormat_Y8_V8_U8
T_U8_Y8_V8_TRUE	UUUUUUUU YYYYYYYY VVVVVVVV	NvColorFormat_U8_Y8_V8
T_V8_Y8_U8_TRUE	VVVVVVVV YYYYYYYY UUUUUUUU	NvColorFormat_V8_Y8_U8
T_U8_V8_Y8_TRUE	UUUUUUUU VVVVVVVV YYYYYYYY	NvColorFormat_U8_V8_Y8
T_V8_U8_Y8_TRUE	VVVVVVVV UUUUUUUU YYYYYYYY	NvColorFormat_V8_U8_Y8
T_R16_G16_B16_I	RRRRRRRRRRRRRRRR GGGGGGGGGGGGGGGG BBBBBBBBBBBBBBBB	NvColorFormat_Int_R16_G16_B16
T_R16_G16_B16_UI	RRRRRRRRRRRRRRRR GGGGGGGGGGGGGGGG BBBBBBBBBBBBBBBB	NvColorFormat_Uint_R16_G16_B16
T_R32_G32_B32	RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB	NvColorFormat_R32_G32_B32
T_R32_G32_B32_I	RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB	NvColorFormat_Int_R32_G32_B32
T_R32_G32_B32_UI	RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB	NvColorFormat_Uint_R32_G32_B32

Single Word Pixel Formats

Single Word type pixel formats, which use a single word to list all components in a single pixel, will list components from left to right, starting from the component in the most significant bit position and ending with the component in the least significant bit position. Little endian byte order is assumed when storing words into memory bytes.

e.g., T_A8B8G8R8

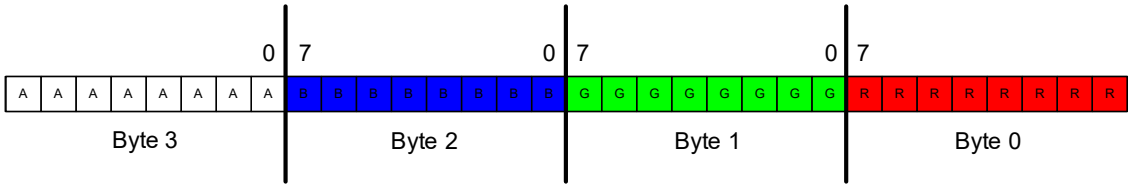


Table 90: Single Word Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name
T_A8	AAAAAAAA	NvColorFormat_A8
T_L8	LLLLLLLL	NvColorFormat_L8
T_L4A4	LLLLAAAA	NvColorFormat_L4A4
T_A4L4	AAAALLLL	NvColorFormat_A4L4
T_G4R4	GGGGRRRR	NvColorFormat_G4R4
T_R8	RRRRRRRR	NvColorFormat_R8
T_R8_SN	RRRRRRRR	NvColorFormat_SNorm_R8
T_R3G3B2	RRRGGGBB	NvColorFormat_R3G3B2
T_A8R8	AAAAAAAARRRRRRRR	NvColorFormat_A8R8
T_A8L8	AAAAAAAALLLLLLLL	NvColorFormat_A8L8
T_L8A8	LLLLLLLLAAAAAAAA	NvColorFormat_L8A8
T_R8G8	RRRRRRRRGGGGGGGG	NvColorFormat_R8G8
T_G8R8	GGGGGGGGRRRRRRRR	NvColorFormat_G8R8
T_G8R8_SN	GGGGGGGGRRRRRRRR	NvColorFormat_SNorm_G8R8
T_R6G5B5	RRRRRRGGGGGBBBBB	NvColorFormat_R6G5B5
T_R5G6B5	RRRRRGGGGGGBBBBB	NvColorFormat_R5G6B5
T_B5G6R5	BBBBBGGGGGGRRRRR	NvColorFormat_B5G6R5
T_R5G5B5A1	RRRRRGGGGGBBBBBA	NvColorFormat_R5G5B5A1
T_B5G5R5A1	BBBBBGGGGGRRRRRA	NvColorFormat_B5G5R5A1
T_A1B5G5R5	ABBBBBGGGGGRRRRR	NvColorFormat_A1B5G5R5
T_A1R5G5B5	ARRRRRGGGGGBBBBB	NvColorFormat_A1R5G5B5
T_R1G5B5A5	RGGGGGBBBBBAAAAA	
T_B5G5R1A5	BBBBBGGGGGRAAAAA	
T_A5B5G5R1	AAAAABBBBBGGGGGR	
T_A5R1G5B5	AAAAARGGGGGBBBBB	
T_R5G5B5X1	RRRRRGGGGGBBBBBX	NvColorFormat_R5G5B5X1
T_B5G5R5X1	BBBBBGGGGGRRRRRX	NvColorFormat_B5G5R5X1
T_X1B5G5R5	XBBBBBGGGGGRRRRR	NvColorFormat_X1B5G5R5
T_X1R5G5B5	XRRRRRGGGGGBBBBB	NvColorFormat_X1R5G5B5
T_R4G4B4A4	RRRRGGGGBBBBAAAA	NvColorFormat_R4G4B4A4
T_B4G4R4A4	BBBBGGGGRRRRAAAA	NvColorFormat_B4G4R4A4
T_A4B4G4R4	AAAABBBBGGGGRRRR	NvColorFormat_A4B4G4R4
T_A4R4G4B4	AAAARRRRGGGGBBBB	NvColorFormat_A4R4G4B4
T_G16R16	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_G16R16
T_G16R16_SN	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_SNorm_G16R16
T_A8B8G8R8	AAAAAAAABBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_A8B8G8R8

Table 90: Single Word Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name
T_A8R8G8B8	AAAAAAAARRRRRRRRGGGGGGGGBBBBBBBB	NvColorFormat_A8R8G8B8
T_B8G8R8A8	BBBBBBBBGGGGGGGGRRRRRRRRAAAAAAAA	NvColorFormat_B8G8R8A8
T_R8G8B8A8	RRRRRRRRGGGGGGGGBBBBBBBBAAAAAAAA	NvColorFormat_R8G8B8A8
T_A8B8G8R8_SN	AAAAAAAABBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_SNorm_A8B8G8R8
T_X8B8G8R8	XXXXXXXXBBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_X8B8G8R8
T_X8R8G8B8	XXXXXXXXRRRRRRRRGGGGGGGGBBBBBBBB	NvColorFormat_X8R8G8B8
T_B8G8R8X8	BBBBBBBBGGGGGGGGRRRRRRRRXXXXXXXX	NvColorFormat_B8G8R8X8
T_R8G8B8X8	RRRRRRRRGGGGGGGGBBBBBBBBXXXXXXXX	NvColorFormat_R8G8B8X8
T_X8B8G8R8_LN	XXXXXXXXBBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_X8B8G8R8
T_X8B8G8R8_SN	XXXXXXXXBBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_SNorm_X8B8G8R8
T_X8R8G8B8_LN	XXXXXXXXRRRRRRRRGGGGGGGGBBBBBBBB	NvColorFormat_X8R8G8B8
T_A2B10G10R10	AABBBBBBBBBBGGGGGGGGGGRRRRRRRRRR	NvColorFormat_A2B10G10R10
T_A2R10G10B10	AARRRRRRRRRRGGGGGGGGGGBBBBBBBBBB	NvColorFormat_A2R10G10B10
T_B10G10R10A2	BBBBBBBBBBGGGGGGGGGGRRRRRRRRRRAA	NvColorFormat_B10G10R10A2
T_R10G10B10A2	RRRRRRRRRRGGGGGGGGGGBBBBBBBBBBAA	NvColorFormat_R10G10B10A2
T_X16B16G16R16	XXXXXXXXXXXXXXXXBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_X16B16G16R16
T_A16B16G16R16	AAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_A16B16G16R16
T_A16	AAAAAAAAAAAAAAAA	NvColorFormat_A16
T_A16_F	AAAAAAAAAAAAAAAA	NvColorFormat_Float_A16
T_L16_F	LLLLLLLLLLLLLLLL	NvColorFormat_Float_L16
T_R16	RRRRRRRRRRRRRRRR	NvColorFormat_R16
T_R16_F	RRRRRRRRRRRRRRRR	NvColorFormat_Float_R16
T_R16_I	RRRRRRRRRRRRRRRR	NvColorFormat_Int_R16
T_R16_SN	RRRRRRRRRRRRRRRR	NvColorFormat_SNorm_R16
T_R16_UI	RRRRRRRRRRRRRRRR	NvColorFormat_Uint_R16
T_A16L16_F	AAAAAAAAAAAAAAAALLLLLLLLLLLLLLLL	NvColorFormat_Float_A16L16
T_A32_F	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	NvColorFormat_Float_A32
T_L32_F	LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL	NvColorFormat_Float_L32
T_R32	RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_ R32
T_R32_F	RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Float_R32
T_G16R16_F	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Float_G16R16
T_E5B9G9R9_F	EEEEEBBBBBBBBBGGGGGGGGGRRRRRRRRR	NvColorFormat_Float_E5B9G9R9
T_B10G11R11_F	BBBBBBBBBBGGGGGGGGGGRRRRRRRRRR	NvColorFormat_Float_B10G11R11
T_R11G11B10_F	RRRRRRRRRRGGGGGGGGGGBBBBBBBBBB	NvColorFormat_Float_R11G11B10
T_A2B10FG10FR10F	AABBBBBBBBBBGGGGGGGGGGRRRRRRRRRR	NvColorFormat_A2B10FG10FR10F
T_X16B16G16R16_F	XXXXXXXXXXXXXXXXBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Float_X16B16G16R16
T_A16B16G16R16_F	AAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Float_A16B16G16R16
T_G32R32_F	GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Float_G32R32
T_X32B32G32R32_F	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Float_X32B32G32R32

Table 90: Single Word Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name
T_A32B32G32R32_F	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Float_A32B32G32R32
T_S8	SSSSSSSS	NvColorFormat_S8
T_Z16	ZZZZZZZZZZZZZZZZ	NvColorFormat_Z16
T_Z16_NL	ZZZZZZZZZZZZZZZZ	NvColorFormat_Z16
T_G24R8	GGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRR	NvColorFormat_G24R8
T_G8R24	GGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_G8R24
T_Z24X8	ZZZZZZZZZZZZZZZZZZZZZZZZXXXXXXXX	NvColorFormat_Z24X8
T_Z24S8	ZZZZZZZZZZZZZZZZZZZZZZZZSSSSSSSS	NvColorFormat_Z24S8
T_X8Z24	XXXXXXXXZZZZZZZZZZZZZZZZZZZZZZZZ	NvColorFormat_X8Z24
T_S8Z24	SSSSSSSSZZZZZZZZZZZZZZZZZZZZZZZZ	NvColorFormat_S8Z24
T_Z32_F	ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ	NvColorFormat_Float_Z32
T_X24S8Z32_F	XXXXXXXXXXXXXXXXXXXXXXXXSSSSSSSSZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ	NvColorFormat_Float_X24S8Z32
T_R8_UI	RRRRRRRR	NvColorFormat_Uint_R8
T_R8_I	RRRRRRRR	NvColorFormat_Int_R8
T_R16_UI	RRRRRRRRRRRRRRRR	NvColorFormat_Uint_R16
T_R16_I	RRRRRRRRRRRRRRRR	NvColorFormat_Int_R16
T_R16_SN	RRRRRRRRRRRRRRRR	NvColorFormat_SNorm_R16
T_G8R8_UI	GGGGGGGGRRRRRRRR	NvColorFormat_Uint_G8R8
T_G8R8_I	GGGGGGGGRRRRRRRR	NvColorFormat_Int_G8R8
T_R32_UI	RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Uint_R32
T_R32_I	RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Int_R32
T_G16R16	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_G16R16
T_G16R16_F	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Float_G16R16
T_G16R16_I	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Int_G16R16
T_G16R16_UI	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Uint_G16R16
T_G16R16_SN	GGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_SNorm_G16R16
T_A16R16	AAAAAAAAAAAAAAAARRRRRRRRRRRRRRRR	NvColorFormat_A16R16
T_A16R16_F	AAAAAAAAAAAAAAAARRRRRRRRRRRRRRRR	NvColorFormat_Float_A16R16
T_A2B10G10R10_UI	AABBBBBBBBBBGGGGGGGGGGRRRRRRRRRR	NvColorFormat_Uint_A2B10G10R10
T_A8B8LNG8LNR8LN	AAAAAAAABBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_A8B8LNG8LNR8LN
T_A8R8LNG8LNB8LN	AAAAAAAARRRRRRRRGGGGGGGGBBBBBBBB	NvColorFormat_A8R8LNG8LNB8LN
T_A8B8G8R8_I	AAAAAAAABBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_Int_A8B8G8R8
T_A8B8G8R8_UI	AAAAAAAABBBBBBBBGGGGGGGGRRRRRRRR	NvColorFormat_Uint_A8B8G8R8
T_G32R32	GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_G32R32
T_G32R32_I	GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Int_G32R32
T_G32R32_UI	GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Uint_G32R32
T_A32R32_F	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAARRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Float_A32R32
T_B24G8R32_F	BBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Float_B24G8R32
T_A16B16G16R16_I	AAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Int_A16B16G16R16

Table 90: Single Word Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name
T_A16B16G16R16_UI	AAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Uint_A16B16G16R16
T_A16B16G16R16_SN	AAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_SNorm_A16B16G16R16
T_X16B16G16R16_I	XXXXXXXXXXXXXXXXBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Int_X16B16G16R16
T_X16B16G16R16_UI	XXXXXXXXXXXXXXXXBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRR	NvColorFormat_Uint_X16B16G16R16
T_A16Y16U16V16_I	AAAAAAAAAAAAAAAAYYYYYYYYYYYYYYYYUUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV	NvColorFormat_Int_A16Y16U16V16
T_V16U16Y16A16_I	VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUUYYYYYYYYYYYYYYYYAAAAAAAAAAAAAAAA	NvColorFormat_Int_V16U16Y16A16
T_A16Y16U16V16_F	AAAAAAAAAAAAAAAAYYYYYYYYYYYYYYYYUUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV	NvColorFormat_Float_A16Y16U16V16
T_A32B32G32R32	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_A32B32G32R32
T_A32B32G32R32_I	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Int_A32B32G32R32
T_A32B32G32R32_UI	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Uint_A32B32G32R32
T_X32B32G32R32_I	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Int_X32B32G32R32
T_X32B32G32R32_UI	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR	NvColorFormat_Uint_X32B32G32R32
T_A8Y8U8V8	AAAAAAAAYYYYYYYYUUUUUUUUVVVVVVVV	NvColorFormat_A8Y8U8V8
T_A8Y8V8U8	AAAAAAAAYYYYYYYYVVVVVVVVUUUUUUUU	NvColorFormat_A8Y8V8U8
T_A8U8V8Y8	AAAAAAAAUUUUUUUUVVVVVVVVYYYYYYYY	NvColorFormat_A8U8V8Y8
T_A8V8U8Y8	AAAAAAAAVVVVVVVVUUUUUUUUYYYYYYYY	NvColorFormat_A8V8U8Y8
T_Y8U8V8A8	YYYYYYYYUUUUUUUUVVVVVVVVAAAAAAAA	NvColorFormat_Y8U8V8A8
T_Y8V8U8A8	YYYYYYYYVVVVVVVVUUUUUUUUAAAAAAAA	NvColorFormat_Y8V8U8A8
T_U8V8Y8A8	UUUUUUUUVVVVVVVVYYYYYYYYAAAAAAAA	NvColorFormat_U8V8Y8A8
T_V8U8Y8A8	VVVVVVVVUUUUUUUUYYYYYYYYAAAAAAAA	NvColorFormat_V8U8Y8A8
T_A8Y8U8V8_TRUE	AAAAAAAAYYYYYYYYUUUUUUUUVVVVVVVV	NvColorFormat_A8Y8U8V8
T_A8Y8V8U8_TRUE	AAAAAAAAYYYYYYYYVVVVVVVVUUUUUUUU	NvColorFormat_A8Y8V8U8
T_A8U8V8Y8_TRUE	AAAAAAAAUUUUUUUUVVVVVVVVYYYYYYYY	NvColorFormat_A8U8V8Y8
T_A8V8U8Y8_TRUE	AAAAAAAAVVVVVVVVUUUUUUUUYYYYYYYY	NvColorFormat_A8V8U8Y8
T_Y8U8V8A8_TRUE	YYYYYYYYUUUUUUUUVVVVVVVVAAAAAAAA	NvColorFormat_Y8U8V8A8
T_Y8V8U8A8_TRUE	YYYYYYYYVVVVVVVVUUUUUUUUAAAAAAAA	NvColorFormat_Y8V8U8A8
T_U8V8Y8A8_TRUE	UUUUUUUUVVVVVVVVYYYYYYYYAAAAAAAA	NvColorFormat_U8V8Y8A8
T_V8U8Y8A8_TRUE	VVVVVVVVUUUUUUUUYYYYYYYYAAAAAAAA	NvColorFormat_V8U8Y8A8
T_U8V8	UUUUUUUUVVVVVVVV	NvColorFormat_U8V8
T_V8U8	VVVVVVVVUUUUUUUU	NvColorFormat_U8_V8
T_U8V8_TRUE	UUUUUUUUVVVVVVVV	NvColorFormat_U8V8

Table 90: Single Word Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name
T_V8U8_TRUE	VVVVVVVVUUUUUUUU	NvColorFormat_U8_V8
T_Y8	YYYYYYYY	NvColorFormat_Y8
T_U8	UUUUUUUU	NvColorFormat_U8
T_V8	VVVVVVVV	NvColorFormat_V8
T_Y8_TRUE	YYYYYYYY	NvColorFormat_Y8
T_U8_TRUE	UUUUUUUU	NvColorFormat_U8
T_V8_TRUE	VVVVVVVV	NvColorFormat_V8
T_P8	PPPPPPPP	NvColorFormat_I8
T_A4P4	AAAAPPPP	NvColorFormat_A4I4
T_A8P8	AAAAAAAAPPPPPPPP	NvColorFormat_A8I8

Miscellaneous Pixel Formats

Table 91: Miscellaneous Pixel Formats

Hardware Name	Bit pattern (Left to Right: LS-Word to MS-Word□ms-bit to ls-bit per Word)	Software Name
T_S8_MSAA	SSSSSSSS	
T_U8_Y8__V8_Y8	UUUUUUUU YYYYYYYY VVVVVVVV YYYYYYYY	NvColorFormat_U8_Y8__V8_Y8 or NvColorFormat_UYVY
T_V8_Y8__U8_Y8	VVVVVVVV YYYYYYYY UUUUUUUU YYYYYYYY	NvColorFormat_V8_Y8__U8_Y8 or NvColorFormat_VYUY
T_Y8_U8__Y8_V8	YYYYYYYY UUUUUUUU YYYYYYYY VVVVVVVV	NvColorFormat_Y8_U8__Y8_V8 or NvColorFormat_YUYV
T_Y8_V8__Y8_U8	YYYYYYYY VVVVVVVV YYYYYYYY UUUUUUUU	NvColorFormat_Y8_V8__Y8_U8 or NvColorFormat_YVYU
T_U8_Y8__V8_Y8_TRUE	UUUUUUUU YYYYYYYY VVVVVVVV YYYYYYYY	NvColorFormat_U8_Y8__V8_Y8
T_V8_Y8__U8_Y8_TRUE	VVVVVVVV YYYYYYYY UUUUUUUU YYYYYYYY	NvColorFormat_V8_Y8__U8_Y8
T_Y8_U8__Y8_V8_TRUE	YYYYYYYY UUUUUUUU YYYYYYYY VVVVVVVV	NvColorFormat_Y8_U8__Y8_V8
T_Y8_V8__Y8_U8_TRUE	YYYYYYYY VVVVVVVV YYYYYYYY UUUUUUUU	NvColorFormat_Y8_V8__Y8_U8
T_B8_G8__R8_G8	BBBBBBBB GGGGGGGG RRRRRRRR GGGGGGGG	NvColorFormat_B8_G8__R8_G8
T_G8_B8__G8_R8	GGGGGGGG BBBBBBBB GGGGGGGG RRRRRRRR	NvColorFormat_G8_B8__G8_R8
T_R1	R	NvColorFormat_R1
T_P1	P	NvColorFormat_I1
T_P2	PP	NvColorFormat_I2
T_P4	PPPP	NvColorFormat_I4
T_DXT1_RGBA		
T_DXT1C_RGB		
T_DXT3_RGBA		
T_DXT5_RGBA		
T_ETC_RGB		
T_ETC3_RGBA		
T_ETC5_RGBA		
T_LATC1		
T_LATC2		
T_ETC2_RGB		
T_ETC2P_RGBA		
T_ETC2A_RGBA		

Table 91: Miscellaneous Pixel Formats

Hardware Name	Bit pattern (Left to Right: LS-Word to MS-Word ms-bit to ls-bit per Word)	Software Name
T_EAC4BPP_R		
T_EAC4BPP_R_SN		
T_EAC4BPP_RG		
T_EAC4BPP_RG_SN		

Multi-Pixel Packing Pixel Formats

Pixel data can come in non-power of two bit sizes that are difficult to fetch and process when contained in certain memory formats (e.g., block linear). For example, raw camera data can come in 6-bit and 10-bit pixel sizes. It can simplify pixel data fetch and processing by packing multiple pixels into a single large word.

Table 92: Multiple Plane Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte ms-bit to ls-bit per Byte)	Software Name
T_X2Lc10Lb10La10	XX $L^2L^2L^2L^2L^2L^2L^2L^2L^2L^2$ $L^1L^1L^1L^1L^1L^1L^1L^1L^1L^1$ $L^0L^0L^0L^0L^0L^0L^0L^0L^0L^0$	
T_X2Le6Ld6Lc6Lb6La6	XX $L^4L^4L^4L^4L^4L^4$ $L^3L^3L^3L^3L^3L^3$ $L^2L^2L^2L^2L^2L^2$ $L^1L^1L^1L^1L^1L^1$ $L^0L^0L^0L^0L^0L^0$	

Multiple Plane Pixel Formats

Some color formats, for example, describing the YUV color space, split pixel data across multiple planes. Fully planar YUV formats contain three planes, one each for Y, U, and V components. Semi-planar YUV formats contain two planes, one for Y, and one shared between U and V components.

Some units may use a different color format name for each plane. Other units may use a single color format name describing all planes in a single name as described below.

Table 93: 8-bit Multiple Plane Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte ms-bit to ls-bit per Byte)	Software Name	FOURCC
T_Y8___U8V8_N444 Y8,U8V8 (YUV444 semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		
T_Y8___V8U8_N444 Y8,V8U8 (YUV444 semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		
T_Y8___U8V8_N422 Y8,U8V8 (YUV422 semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		NV61
T_Y8___V8U8_N422 Y8,V8U8 (YUV422 semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		NV16
T_Y8___U8V8_N422R Y8,U8V8 (YUV422R semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		
T_Y8___V8U8_N422R Y8,V8U8 (YUV422R semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		
T_Y8___U8V8_N420 Y8,U8V8 (YUV420 semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		NV21
T_Y8___V8U8_N420 Y8,V8U8 (YUV420 semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		NV12
T_Y8___U8___V8_N444 Y8,U8, V8 (YUV444 full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		YV24

Table 93: 8-bit Multiple Plane Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name	FOURCC
T_Y8___U8___V8_N422 Y8,U8, V8 (YUV422 full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		YV16
T_Y8___U8___V8_N422R Y8,U8, V8 (YUV422R full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		
T_Y8___U8___V8_N420 Y8,U8, V8 (YUV420 full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		YV12
T_Y8___U8V8_N444_TRUE Y8,U8V8 (YUV444 semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		
T_Y8___V8U8_N444_TRUE Y8,V8U8 (YUV444 semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		
T_Y8___U8V8_N422_TRUE Y8,U8V8 (YUV422 semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		
T_Y8___V8U8_N422_TRUE Y8,V8U8 (YUV422 semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		
T_Y8___U8V8_N422R_TRUE Y8,U8V8 (YUV422R semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		
T_Y8___V8U8_N422R_TRUE Y8,V8U8 (YUV422R semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		
T_Y8___U8V8_N420_TRUE Y8,U8V8 (YUV420 semi-planar)	Plane0: YYYYYYYY Plane1: UUUUUUUUVVVVVVVV		
T_Y8___V8U8_N420_TRUE Y8,V8U8 (YUV420 semi-planar)	Plane0: YYYYYYYY Plane1: VVVVVVVVUUUUUUUU		
T_Y8___U8___V8_N444_TRUE Y8,U8, V8 (YUV444 full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		
T_Y8___U8___V8_N422_TRUE Y8,U8, V8 (YUV422 full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		
T_Y8___U8___V8_N422R_TRUE Y8,U8, V8 (YUV422R full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		
T_Y8___U8___V8_N420_TRUE Y8,U8, V8 (YUV420 full planar)	Plane0: YYYYYYYY Plane1: UUUUUUUU Plane2: VVVVVVVV		

Table 94: 10-bit Multiple Plane Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name	FOURCC
T_Y10___U10V10_N444 Y10,U10V10 (YUV444 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		
T_Y10___V10U10_N444 Y10,V10U10 (YUV444 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVVV000000UUUUUUUUUU000000		
T_Y10___U10V10_N422 Y10,U10V10 (YUV422 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		NV61
T_Y10___V10U10_N422 Y10,V10U10 (YUV422 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVVV000000UUUUUUUUUU000000		NV16

Table 94: 10-bit Multiple Plane Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name	FOURCC
T_Y10___U10V10_N422R Y10,U10V10 (YUV422R semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		
T_Y10___V10U10_N422R Y10,V10U10 (YUV422R semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVVV000000UUUUUUUUUU000000		
T_Y10___U10V10_N420 Y10,U10V10 (YUV420 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		NV21
T_Y10___V10U10_N420 Y10,V10U10 (YUV420 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVVV000000UUUUUUUUUU000000		NV12
T_Y10___U10___V10_N444 Y10,U10, V10 (YUV444 full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		YV24
T_Y10___U10___V10_N422 Y10,U10, V10 (YUV422 full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		YV16
T_Y10___U10___V10_N422R Y10,U10, V10 (YUV422R full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		
T_Y10___U10___V10_N420 Y10,U10, V10 (YUV420 full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		YV12
T_Y10___U10V10_N444_TRUE Y10,U10V10 (YUV444 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		
T_Y10___V10U10_N444_TRUE Y10,V10U10 (YUV444 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVVV000000UUUUUUUUUU000000		
T_Y10___U10V10_N422_TRUE Y10,U10V10 (YUV422 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		
T_Y10___V10U10_N422_TRUE Y10,V10U10 (YUV422 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVVV000000UUUUUUUUUU000000		
T_Y10___U10V10_N422R_TRUE Y10,U10V10 (YUV422R semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		
T_Y10___V10U10_N422R_TRUE Y10,V10U10 (YUV422R semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVVV000000UUUUUUUUUU000000		
T_Y10___U10V10_N420_TRUE Y10,U10V10 (YUV420 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000VVVVVVVVVV000000		
T_Y10___V10U10_N420_TRUE Y10,V10U10 (YUV420 semi-planar)	Plane0: YYYYYYYYYY000000 Plane1: VVVVVVVVV000000VUUUUUUUUUU000000		
T_Y10___U10___V10_N444_TRUE Y10,U10, V10 (YUV444 full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		
T_Y10___U10___V10_N422_TRUE Y10,U10, V10 (YUV422 full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		
T_Y10___U10___V10_N422R_TRUE Y10,U10, V10 (YUV422R full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		
T_Y10___U10___V10_N420_TRUE Y10,U10, V10 (YUV420 full planar)	Plane0: YYYYYYYYYY000000 Plane1: UUUUUUUUUU000000 Plane2: VVVVVVVVVV000000		

Table 95: 16-bit Multiple Plane Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name	FOURCC
T_Y16___U16V16_N444 Y16,U16V16 (YUV444 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		
T_Y16___V16U16_N444 Y16,V16U16 (YUV444 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		
T_Y16___U16V16_N422 Y16,U16V16 (YUV422 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		NV61
T_Y16___V16U16_N422 Y16,V16U16 (YUV422 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		NV16
T_Y16___U16V16_N422R Y16,U16V16 (YUV422R semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		
T_Y16___V16U16_N422R Y16,V16U16 (YUV422R semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		
T_Y16___U16V16_N420 Y16,U16V16 (YUV420 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		NV21
T_Y16___V16U16_N420 Y16,V16U16 (YUV420 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		NV12
T_Y16___U16___V16_N444 Y16,U16, V16 (YUV444 full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		YV24
T_Y16___U16___V16_N422 Y16,U16, V16 (YUV422 full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		YV16
T_Y16___U16___V16_N422R Y16,U16, V16 (YUV422R full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		
T_Y16___U16___V16_N420 Y16,U16, V16 (YUV420 full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		YV16
T_Y16___U16V16_N444_TRUE Y16,U16V16 (YUV444 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		
T_Y16___V16U16_N444_TRUE Y16,V16U16 (YUV444 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		
T_Y16___U16V16_N422_TRUE Y16,U16V16 (YUV422 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		
T_Y16___V16U16_N422_TRUE Y16,V16U16 (YUV422 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		
T_Y16___U16V16_N422R_TRUE Y16,U16V16 (YUV422R semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		
T_Y16___V16U16_N422R_TRUE Y16,V16U16 (YUV422R semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		
T_Y16___U16V16_N420_TRUE Y16,U16V16 (YUV420 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUUVVVVVVVVVVVVVVVV		
T_Y16___V16U16_N420_TRUE Y16,V16U16 (YUV420 semi-planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: VVVVVVVVVVVVVVVVUUUUUUUUUUUUUUUU		
T_Y16___U16___V16_N444_TRUE Y16,U16, V16 (YUV444 full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		
T_Y16___U16___V16_N422_TRUE Y16,U16, V16 (YUV422 full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		

Table 95: 16-bit Multiple Plane Pixel Formats

Hardware Name	Bit pattern (Left to Right: MS-Byte to LS-Byte□ms-bit to ls-bit per Byte)	Software Name	FOURCC
T_Y16___U16___V16_N422R_TRUE Y16,U16, V16 (YUV422R full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		
T_Y16___U16___V16_N420_TRUE Y16,U16, V16 (YUV420 full planar)	Plane0: YYYYYYYYYYYYYYYY Plane1: UUUUUUUUUUUUUUUU Plane2: VVVVVVVVVVVVVVVV		

20.4.1.3 Bayer Pixel Formats

Bayer pixel data can come from camera sensor equipment in various sizes, e.g., 6bits, 10bits, 14bits, 16bits, etc. Most Bayer data is stored using existing pixel formats, such as T_L8, or T_L16. In some special cases, special formats are created to pack pixel data efficiently to work with existing memory formats, e.g., T_X2Lc10Lb10La10, T_X2Le6Ld6Lc6Lb6La6.

20.4.1.4 Maxwell Caveats

Pixel formats that are not a power of 2 bits cannot be used together with block linear memory tiling formats because they cannot fit into a block linear GOB (256 or 512 bytes) without additional padding. These formats can only be used with pitch linear surfaces.

The list of pixel formats that will not work with block linear memory tiling is listed below:

- 24 bit packed RGB
  - T_R8_G8_B8, T_B8_G8_R8, T_R8_G8_B8_SN, T_R8_G8_B8_UI, T_R8_G8_B8_I)
- 24bit packed YUV
  - i.e., T_Y8_U8_V8, T_Y8_V8_U8, T_U8_Y8_V8, T_V8_Y8_U8, T_U8_V8_Y8, T_V8_U8_Y8
- T_R16_G16_B16_I, T_R16_G16_B16_UI
- T_R32_G32_B32_I, T_R32_G32_B32_UI

20.5 Compression Overview

Tegra employs lossless surface compression without compaction on 4BPP ARGB surface data. Tegra uses GPU compression techniques whereby surfaces are compressed in tiled regions. This allows for bandwidth reduction but the surface footprint remains uncompressed.

Techniques for compression vary based on the surface type. Compression techniques these surface types are:

- Arithmetic
  - 2:1 surface compression
  - Improved algorithm for Maxwell utilizing Variable Difference Compression (VDC)
- Reduction
  - 4:1 or 8:1 surface compression
  - Efficient for pixel replication within a tiled region
- Zero Bandwidth Clear (ZBC)
  - Surface tile made up of single color
  - Only 1 ZBC color per surface
  - No memory access required

Tegra X1 supports two forms of sRGB surface compression:

- 1xAA support using 2CRA surface compression
  - ZBC
  - 8:1 or 4:1 reduction
  - 2:1 VDC
  - Uncompressed
- 4xAA support using BRA surface compression
  - 8:1 compression
  - 4:1 reduction followed by 2:1 VDC
  - 4:1 reduction
  - 2:1 VDC
  - Uncompressed

The surface tile follows the GPU ROP tile dimension of 32Bx8 or 256B or half of a GOB. The compressed data is stored in this tile as:

- 32 B (8:1 compression ratio)
- 64 B (4:1 compression ratio)
- 128 B (2:1 compression ratio)
- 256 B (1:1 compression ratio)

20.5.1 Compression Format Kinds

Tegra X1 compression formats are taken from GPU and therefore follow GPU compression format naming conventions. Tegra X1 will support the following compression kind for RGB:

- RGB
  - NV_MMU_PTE_KIND_C32_2CRA (2CRA)
  - NV_MMU_PTE_KIND_C32_MS4_2BRA (2BRA)

The compression kind for GPU 1xAA surfaces is 2CRA and will be used for the following pixel formats:

- 32-bit ARGB

The compression kind for GPU 4xAA surfaces is 2BRA and will be used for the following pixel formats:

- 32-bit ARGB

20.5.2 Compression Format Mappings

Compression tags

Description of 2CRA compression mapping:

- Uncompressed enum CROP_CS_2CRA_U0
- Zero Bandwidth Clear enum CROP_CS_2CRA_C 1
- Reduction Compressed enum CROP_CS_2CRA_R 2
- Arithmetic Compressed enum CROP_CS_2CRA_A 3

Description of 2BRA compression mapping:

- Uncompressed enum CROP_CS_2BRA_U0
- Both Compressed enum CROP_CS_2BRA_B 1

- Reduction Compressed enum CROP_CS_2BRA_R 2
- Arithmetic Compressed enum CROP_CS_2BRA_A 3

20.6 End-to-End Support

20.6.1 Overview

To determine if Tegra X1 can support a given multimedia use case, it is important to understand the full path from a system point of view. Limits on important parameters associated with the use case must be documented to determine if the use case is possible. For example, the maximum resolution associated with H.264 video stream. The intent of this section is to associate those parameters with pixel memory formats to determine if Tegra X1 can support a given use case. Important parameters associated with these formats are:

- Codec
- Color space/component depth
- Resolution
- Frame rate

The pixel processing paths described in this section are:

- Video input path
- Video encode path
- Camera input path
- Display output path

Analysis of the end to end path is broken down into three relatively orthogonal sections:

- Input
- Internal
- Output

Each section is meant to capture the extents of support provided on the input path, internally or out to a display panel. Ideally this information would be incorporated into the use case model (UCM) so that along with bandwidth limitations, those limitations captured in the following sections would also be checked to determine IMP (is-mode-possible) for a given set of use cases.

20.6.2 Input Support

Tegra X1 has two major pixel capture paths: video and camera input. The video input path is processed by NVDEC which supports a variety of CODECs. The camera input path is supported by VI and ISP blocks.

The table below shows the extent of support on the video input path. Each CODEC has a list of parameters describing the limit of support for that CODEC. The color space and maximum component depth describe the extents that YUV data is captured for that CODEC.

Notes:

- *Color space is not manipulated in NVDEC so only the component depth determines color space support for the input video path.*
- *ULV refers to Ultra Low Voltage or minimum voltage level allowed for Tegra X1.*
- *SV refers to Standard Voltage which is actually the maximum voltage used for Tegra X1 performance targets.*

Table 96: Video Decode Input Path (NVDEC)

CODEC	FORMAT	COLOR SPACE	MAX DEPTH	MAX RES	ULV Frame Rate	SV Frame Rate
H.264, MPEG2 VP8	YUV420	Rec.601 Rec.709	8 bpc	4096x4096	4K at 30fps	4K at 60fps
VC1 MPEG4	YUV420	Rec.601 Rec.709	8 bpc	2048x1024	1080p at 120fps	1080p at 120fps
H.265	YUV420, YUV444	Rec.601 Rec.709 Rec.2020	8/10/12 bpc (yuv444 don't support 12bpc)	4096x4096	4K at 30fps	4K at 60fps
VP9	YUV420	Rec.601 Rec.709 Rec.2020	8/10/12 bpc	4096x4096	4K at 30fps	4K at 60fps

The camera input path has major points that can impose limits on pixel throughput:

- CSI
  - 1 or 2 sensors (x4) – tablet platform
    - 12 bpp
    - 30 MPix/frame @ 30fps
    - 900 MPix/s (single sensor)
    - 1.8 GPix/s (dual sensor)
  - 12 sensors – automotive platform
    - 16 bpp
    - 12 * 4 MPix/frame @ 30fps
    - 1.44 GPix/s
  - Maximum sensor resolution of 50MPel
  - Maximum pixel depth of 16 bpp RAW format
- ISP
  - 2 pixel per clock
  - 700 MHz clock at SV
  - 1.4 GPix/s
  - Independent of pixel depth
  - Maximum component depth of 12 bpc
  - Output format of YUV420, YUV444

Using this information, the table below describes the camera capture rate of various sensor resolutions as well as for a 12-sensor automotive platform.

Note: *ISP supports a maximum of 1.4 GPix/s throughput regardless of YUV format or component depth. Therefore ignoring memory bandwidth, camera input path can output 12 bpc Rec.2020 output at the same rate as 8 bpc Rec.601.*

Table 97: Camera Input Path (Maximum Performance)

Sensor Configuration			CSI Capture Rate (fps)	ISP Capture Rate (fps)	Camera Path Capture Rate (fps)
Number Sensors	Sensor Resolution	Sensor Connection			
1	8 MPEL	X4 12bpp	112	175	112
1	13 MPEL	X4 12bpp	69	112	69
1	20 MPEL	X4 12bpp	45	70	45
1	50 MPEL	X4 12bpp	18	28	18
2	8 MPEL	X4 12bpp	224	175	175
2	13 MPEL	X4 12bpp	138	112	112
2	20 MPEL	X4 12bpp	90	70	70
2	50 MPEL	X4 12bpp	36	28	28
12	4 MPEL	All lanes 16bpp	30	29	29

20.6.3 Internal Support

Internal support refers to those blocks downstream from the input path but prior to prior to panel or video encode blocks. Internal path processing blocks include:

- GPU
- VIC
- Display pipeline
- ISP (image improvement stages)

The pixel processing functions provided in these blocks are necessary for areas such as:

- Camera image quality
- Video post processing
- Color space conversion
- Scaling
- Compositing

This section focuses on the parameter that must be carried through the internal blocks and the effects on the parameter due to processing limitations of the internal blocks, namely color space. The good news is Tegra X1 provides full support for Rec.601 and Rec.709 color spaces as well as 10bpc Rec.2020 color space. The bad news is 12bpc Rec.2020 color space images may suffer precision loss when images are processed by VIC or display.

VIC and display have internal pipeline precision of S1.14 which provides an acceptable level of precision for blending 8-bit and 10-bit surfaces. For scaling and blending 12-bit surfaces, an internal pipeline color format of S1.16 is required. Therefore 12-bit content that requires scaling or blending will suffer precision loss. This issue could be noticed in systems with a 12-bit output panel. Systems with lower component depth panels should not be able to perceive the drop in precision. The precision of 12-bit video content that do not require gamma operations, scaling or blending will be maintained. Note that GPU can avoid precision loss by compositing using 32-bpc floating point processing. GPU outputs can then be converted to 16 bpc UNORM ARGB outputs.

Bandwidth limitations affecting refresh rates are beyond the scope of this document.

Note: *Internal blocks all support resolutions of 16K x 16K. VIC and display support 8K x 8K for reading compressed images.*

20.6.4 Output Support

There are two outputs covered in this section:

- Video encode path (NVENC)
- Display panel support

The video encode path limitations are described below.

Table 98: Video Encode Path (NVENC)

CODEC	FORMAT	COLOR SPACE	MAX DEPTH	MAX RES	SV Frame Rate
VC1, MPEG2, MPEG4, VP8, VP9	YUV420	Rec.601, Rec.709	8 bpc	4096x4096	4K at 30fps
H.264	YUV420, YUV444	Rec.601, Rec.709	8 bpc		YUV420: 4K at 60fps YUV444: 4K at 30fps
H.265	YUV420, YUV444	Rec.601, Rec.709	8 bpc		

The following tables show the component depth supported for a given output panel resolution. Limitations are based on the available bandwidth for a given panel's link speed as well as link format support. It should be noted that display has the flexibility to support color space conversion to and from any of the supported color spaces for Tegra X1.

Namely:

- sRGB
- Rec.601
- Rec.709
- Rec.2020

Therefore display can support any color space for a given resolution within the display pipeline. However Rec.2020 minimum component depth is 10 bits. Therefore, any output unable to generate 10-bpc or greater cannot claim Rec.2020 support.

Table 99: eDP Maximum Output Color Depth

Resolution	Maximum Component Depth Supported			
	RGB Format	YUV420 Format[1]	YUV422 Format	YUV444 Format
Up to 2560x1440@120Hz	10 bpc	See note 1	8 bpc	10 bpc
Up to 3840x2160@30Hz	12 bpc	See note 1	8 bpc	12 bpc
Up to 3840x2160@60Hz	10 bpc	See note 1	8 bpc	10 bpc
Up to 5120x3200@60Hz	Unsupported	See note 1	8 bpc	Unsupported

Table 100: HDMI Output Color Depth

Resolution	Component Depth Supported			
	RGB Format	YUV420 Format[1]	YUV422 Format	YUV444 Format
Up to 2560x1440@120Hz	8 and 12 bpc	See note	8, 10 and 12 bpc	8 and 12 bpc
Up to 3840x2160@30Hz	8 and 12 bpc bpc	See note	8, 10 and 12 bpc	8 and 12 bpc
Up to 3840x2160@60Hz	8 bpc	See note	8 bpc	8 bpc
Up to 5120x3200@60Hz	Unsupported	See note	Unsupported	Unsupported

Table 101: DSI Maximum Output Color Depth

Resolution	Maximum Component Depth Supported			
	RGB Format	YUV420 Format	YUV422 Format	YUV444 Format
Up to 3840x2160@30Hz	8 bpc	Unsupported	Unsupported	Unsupported
Up to 3840x2160@60Hz	8 bpc	Unsupported	Unsupported	Unsupported
Up to 5120x3200@60Hz	8 [2] bpc	Unsupported	Unsupported	Unsupported

Notes:

- *YUV420 is not a supported output of display. WAR is to use a single-surface pass-through mode where an HDMI 8bpc YUV420 surface generated via a GPU shader can be sent directly to the panel through the display pipeline. Note that display can convert YUV420 to YUV422 or YUV444 on the fly and send YUV422 or YUV444 to the monitor.*
- *Support for RGB and YUV444 formats require DSI output compression to achieve the refresh rates for the resolutions shown.*
- *Unlike eDP and DSI color depth support tables, HDMI only color depths listed in the table are supported (i.e., 10 bpc depths are not supported for RGB and YUV444).*

CHAPTER 21: APB

21.1 APB Miscellaneous Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

This chapter describes a number of system control registers that are grouped together in the aptly named miscellaneous registers section. These registers are all present on the APB bus.

21.1.1 APB Control Registers

21.1.1.1 APB_MISC_PP_STRAPPING_OPT_A_0

Strapping Options Register

Offset: 0x8 | Read/Write: R/W | Reset: 0xXX00XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxx00xxxxxxx0)

Bit	Reset	Description
28:26	X	BOOT_SELECT: read at power-on reset, main reset, tsensor reset, and watchdog timer reset time from uart3_txd_IB, uart4_txd_IB, uart4_rts_IB
13	X	NVPROD_UART
12:10	X	RCM_STRAPS: read at power-on reset, main reset, tsensor reset, and watchdog timer reset time from button_home_IB, button_vol_down_IB, button_vol_up_IB
9	0x0	BOOT_FAST_UART: UART Boot speed from TMC JTAG configuration bit 0=57600 baud, 1=Osc Frequency (1 bit per OSC clock) 0 = SLOW 1 = FAST
8	RSVD1	MIO_WIDTH: 0 = RSVD1 1 = RSVD2
5:4	X	RAM_CODE: read at power-on reset, main reset, tsensor reset, and watchdog timer reset time from uart1_tx_IB, uart1_rts_IB
0	RSVD1	NOR_WIDTH: 0 = RSVD1 1 = RSVD2

21.1.1.2 APB_MISC_PP_CONFIG_CTL_0

Configuration Control Register

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxx00)

Bit	Reset	Description
7	DISABLE	TBE: 0 = Disable; 1 = Enable RTCK Daisy chaining 0 = DISABLE 1 = ENABLE
1	DISABLE	XBAR_SO_DEFAULT: Deprecated -- keep disabled 0 = DISABLE 1 = ENABLE
0	DISABLE	CPU_XBAR_SO_ENABLE: Deprecated -- keep disabled 0 = DISABLE 1 = ENABLE

21.1.1.3 APB_MISC_PP_PINMUX_GLOBAL_0_0

Global Pinmux Control Register

When both CLAMP_INPUTS_WHEN_TRISTATED is set to ENABLE and the TRISTATE field is set to TRISTATE (1b1), the inputs to the core are clamped to zero. Software must set this bit before taking any controller using a pinmuxed pin out of reset.

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	CLAMP_INPUTS_WHEN_TRISTATED: 0 = Do not clamp inputs when tristated; 1 = Clamp inputs when tristated 0 = DISABLE 1 = ENABLE

21.1.1.4 APB_MISC_SC1X_PADS_VIP_VCLKCTRL_0

VCLK Control Register

Enable VCLK pad to get external clock for VI.

Offset: 0x428 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	INVERSION: VCLK invert enable 0 = DISABLE 1 = ENABLE
0	0x0	IE: VCLK input enable 0 = DISABLE 1 = ENABLE

21.1.1.5 APB_MISC_GP_HIDREV_0

Chip ID Revision Register

Offset: 0x804 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	MINORREV: Chip ID minor revision. Minor revisions from 2 to 10 are reserved.
15:8	X	CHIPID: Chip ID
7:4	X	MAJORREV: Chip ID major revision (0: Emulation, 1-15: Silicon) 0 = EMULATION 1 = A01
3:0	X	HIDFAM: Chip ID family register. 0 = GPU 1 = HANDHELD 2 = BR_CHIPS 3 = CRUSH 4 = MCP 5 = CK 6 = VAIO 7 = HANDHELD_SOC

21.1.2 Pad Control Registers

The registers below control pad behavior. For each digital pad, the following controls can be used to tune pad performance, functionality, and power consumption. The pads controlled by each pad group register can be found in Chapter 9: Multi-Purpose I/O Pins and Pin Multiplexing (Pinmuxing) of this document.

- **HSM_EN** - High speed mode - active high, enables high speed mode for driver and receiver for better matching of the rise/fall delay in outbound and inbound paths. Use it for clocks and the high speed signaling where the matching timings are of importance.

- SCHMT_EN - Schmitt enable - active high, enables the Schmitt Trigger Type of I/P receiver. Default is Inverter Type of receiver.
- CAL_DRVDN - drive down (falling edge) - Driver Output Pull-Down drive strength code.
- CAL_DRVUP - drive up (rising edge) - Driver Output Pull-Up drive strength code.
- DRVDN_SLWR - Driver Output Rising Edge Slew 2-bit control code.

  Code 11 is least slewing of signal, code 00 is highest slewing of the signal.
- DRVUP_SLWF -Driver Output Falling Edge Slew 2-bit control code.

  Code 11 is least slewing of signal, code 00 is highest slewing of the signal.

21.1.3 APB_MISC_GP_ASDBGREG_0

Debug registers. Has controls for debug enable, and performance enable.

Offset: 0x810 I Read/Write: R/W I Reset: 0x00000000 (0bxx0000000000xxxxxxxxxxxx00xxx00x)

Bit	Reset	Description
29:28	0x0	CFG2TMC_RAM_SVOP_SP: control write timing characteristics for the compiled RAMSP, add reset to RAM_SVOP_SP
27:26	0x0	CFG2TMC_RAM_SVOP_REG: control write timing characteristics for the compiled RAMREG
25:24	0x0	CFG2TMC_RAM_SVOP_PDP: control write timing characteristics for the compiled RAMPDP[a]
23:22	0x0	CFG2TMC_RAM_SVOP_DP: control write timing characteristics for the compiled RAMDP
21:20	0x0	CFG2TMC_RAM_EMAA: control timing characteristics for the compiled rams 0 = DISABLE 1 = ENABLE
7:6	0x0	CFG2TMC_CLKBYP_FUNC:8 rw CFG2TMC_SW_BP_T1CLK i=0x0 enum (DISABLE, ENABLE) 9 rw CFG2TMC_SW_BP_T2CLK i=0x0 enum (DISABLE, ENABLE)10 rw CFG2TMC_SW_BP_T3CLK i=0x0 enum (DISABLE, ENABLE)11 rw CFG2TMC_SW_BP_T4CLK i=0x0 enum (DISABLE, ENABLE)12 rw CFG2TMC_SW_BP_T5CLK i=0x0 enum (DISABLE, ENABLE)13 rw CFG2TMC_SW_BP_T6CLK i=0x0 enum (DISABLE, ENABLE)14 rw CFG2TMC_SW_BP_T7CLK i=0x0 enum (DISABLE, ENABLE)15 rw CFG2TMC_SW_BP_CLK_DIV i=0x0 enum (DISABLE, ENABLE) 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_PULLDOWN_EN: Enables pulldown 0 = DISABLE 1 = ENABLE
1	0x0	CFG2TMC_PULLUP_EN: Enables pullup 0 = DISABLE 1 = ENABLE
0	0x0	CFG2TMC_IDDQ_EN: Enables iddq WARNING: Will functionally kill chip) enum 0 = DISABLE 1 = ENABLE

a. Software needs to program CFG2TMC_RAM_SVOP_PDP to 0x2 as part of cold and warm boot initialization.

21.1.4 APB_MISC_GP_SDMMC1_CLK_LPBK_CONTROL_0

This bit is set when pads are used for SDMMC/eMMC, otherwise it is reset.

Offset: 0x8d4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	SDMMC1_CLK_PAD_E_LPBK: when set, enables deep loopback in SDMMC1 CLK pad

21.1.4.1 APB_MISC_GP_SDMMC3_CLK_LPBK_CONTROL_0

Offset: 0x8d8 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	SDMMC3_CLK_PAD_E_LPBK: When set, enables deep loopback in SDMMC3 CLK pad

21.1.4.2 APB_MISC_GP_EMMC2_PAD_CFG_CONTROL_0

Offset: 0x8dc | Read/Write: R/W | Reset: 0x0000ff37 (0bxxxxxxxxxxxxxxxx11111111xx11x111)

Bit	Reset	Description
15:8	0xff	EMMC2_PAD_E_INPUT: Enable input path for D0-D7; enable=1, disable=0
5	0x1	EMMC2_PAD_E_INPUT_CLK: Enable input for CLk pad. Always on by default.
4	0x1	EMMC2_PAD_E_INPUT_CLKB: Enable input for CLkB pad. Always on by default.
2	0x1	EMMC2_PAD_E_INPUT_DQS: Enable Differential receiver for the DQS
1	0x1	EMMC2_PAD_E_INPUT_DQSB: Enable Differential receiver for the DQSB
0	0x1	EMMC2_PAD_E_DEEP_LPBK_CLK: When set, enables deep loopback in SDMMC2 CLK pad; Should be set to 1 to get loopback clock for capturing read data from device

21.1.4.3 APB_MISC_GP_EMMC4_PAD_CFG_CONTROL_0

Offset: 0x8e0 | Read/Write: R/W | Reset: 0x0000ff37 (0bxxxxxxxxxxxxxxxx11111111xx11x111)

Bit	Reset	Description
15:8	0xff	EMMC4_PAD_E_INPUT: Enable input path for D0-D7; enable=1, disable=0
5	0x1	EMMC4_PAD_E_INPUT_CLK: Enable input for CLK pad. Always on by default.
4	0x1	EMMC4_PAD_E_INPUT_CLKB: Enable input for CLKB pad. Always on by default.
2	0x1	EMMC4_PAD_E_INPUT_DQS: Enable Differential receiver for the DQS
1	0x1	EMMC4_PAD_E_INPUT_DQSB: Enable Differential receiver for the DQSB
0	0x1	EMMC4_PAD_E_DEEP_LPBK_CLK: When set, enables deep loopback in SDMMC4 CLK pad; Should be set to 1 to get loopback clock for capturing read data from device.

21.1.4.4 APB_MISC_GP_ALS_PROX_INT_CFGPADCTRL_0

ALS_PROX_INT_CFG Pad Control Register

Offset: 0x8e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_ALS_PROX_INT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_ALS_PROX_INT_CFG_CAL_DRVDN

21.1.4.5 APB_MISC_GP_AP_READY_CFGPADCTRL_0

AP_READY_CFG Pad Control Register

Offset: 0x8e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_AP_READY_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_AP_READY_CFG_CAL_DRVDN

21.1.4.6 APB_MISC_GP_AP_WAKE_BT_CFGPADCTRL_0

AP_WAKE_BT_CFG Pad Control Register

Offset: 0x8ec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_AP_WAKE_BT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_AP_WAKE_BT_CFG_CAL_DRVDN

21.1.4.7 APB_MISC_GP_AP_WAKE_NFC_CFGPADCTRL_0

AP_WAKE_NFC_CFG Pad Control Register

Offset: 0x8f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_AP_WAKE_NFC_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_AP_WAKE_NFC_CFG_CAL_DRVDN

21.1.4.8 APB_MISC_GP_AUD_MCLK_CFGPADCTRL_0

AUD_MCLK_CFG Pad Control Register

Offset: 0x8f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_AUD_MCLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_AUD_MCLK_CFG_CAL_DRVDN

21.1.4.9 APB_MISC_GP_BATT_BCL_CFGPADCTRL_0

BATT_BCL_CFG Pad control register

Offset: 0x8f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BATT_BCL_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BATT_BCL_CFG_CAL_DRVDN

21.1.4.10 APB_MISC_GP_BT_RST_CFGPADCTRL_0

BT_RST_CFG Pad control register

Offset: 0x8fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BT_RST_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BT_RST_CFG_CAL_DRVDN

21.1.4.11 APB_MISC_GP_BT_WAKE_AP_CFGPADCTRL_0

BT_WAKE_AP_CFG Pad control register

Offset: 0x900 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BT_WAKE_AP_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BT_WAKE_AP_CFG_CAL_DRVDN

21.1.4.12 APB_MISC_GP_BUTTON_HOME_CFGPADCTRL_0

BUTTON_HOME_CFG Pad control register

Offset: 0x904 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BUTTON_HOME_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BUTTON_HOME_CFG_CAL_DRVDN

21.1.4.13 APB_MISC_GP_BUTTON_POWER_ON_CFGPADCTRL_0

BUTTON_POWER_ON_CFG Pad control register

Offset: 0x908 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BUTTON_POWER_ON_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BUTTON_POWER_ON_CFG_CAL_DRVDN

21.1.4.14 APB_MISC_GP_BUTTON_SLIDE_SW_CFGPADCTRL_0

BUTTON_SLIDE_SW_CFG Pad control register

Offset: 0x90c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BUTTON_SLIDE_SW_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BUTTON_SLIDE_SW_CFG_CAL_DRVDN

21.1.4.15 APB_MISC_GP_BUTTON_VOL_DOWN_CFGPADCTRL_0

BUTTON_VOL_DOWN_CFG Pad control register

Offset: 0x910 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BUTTON_VOL_DOWN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BUTTON_VOL_DOWN_CFG_CAL_DRVDN

21.1.4.16 APB_MISC_GP_BUTTON_VOL_UP_CFGPADCTRL_0

BUTTON_VOL_UP_CFG Pad control register

Offset: 0x914 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_BUTTON_VOL_UP_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_BUTTON_VOL_UP_CFG_CAL_DRVDN

21.1.4.17 APB_MISC_GP_CAM1_MCLK_CFGPADCTRL_0

CAM1_MCLK_CFG Pad control register

Offset: 0x918 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM1_MCLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM1_MCLK_CFG_CAL_DRVDN

21.1.4.18 APB_MISC_GP_CAM1_PWDN_CFGPADCTRL_0

CAM1_PWDN_CFG Pad control register

Offset: 0x91c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM1_PWDN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM1_PWDN_CFG_CAL_DRVDN

21.1.4.19 APB_MISC_GP_CAM1_STROBE_CFGPADCTRL_0

CAM1_STROBE_CFG Pad control register

Offset: 0x920 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM1_STROBE_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM1_STROBE_CFG_CAL_DRVDN

21.1.4.20 APB_MISC_GP_CAM2_MCLK_CFGPADCTRL_0

CAM2_MCLK_CFG Pad control register

Offset: 0x924 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM2_MCLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM2_MCLK_CFG_CAL_DRVDN

21.1.4.21 APB_MISC_GP_CAM2_PWDN_CFGPADCTRL_0

CAM2_PWDN_CFG Pad control register

Offset: 0x928 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM2_PWDN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM2_PWDN_CFG_CAL_DRVDN

21.1.4.22 APB_MISC_GP_CAM_AF_EN_CFGPADCTRL_0

CAM_AF_EN_CFG Pad control register

Offset: 0x92c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM_AF_EN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM_AF_EN_CFG_CAL_DRVDN

21.1.4.23 APB_MISC_GP_CAM_FLASH_EN_CFGPADCTRL_0

CAM_FLASH_EN_CFG Pad control register

Offset: 0x930 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM_FLASH_EN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM_FLASH_EN_CFG_CAL_DRVDN

21.1.4.24 APB_MISC_GP_CAM_I2C_SCL_CFGPADCTRL_0

CAM_I2C_SCL_CFG Pad control register

Offset: 0x934 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM_I2C_SCL_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM_I2C_SCL_CFG_CAL_DRVDN

21.1.4.25 APB_MISC_GP_CAM_I2C_SDA_CFGPADCTRL_0

CAM_I2C_SDA_CFG Pad control register

Offset: 0x938 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM_I2C_SDA_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM_I2C_SDA_CFG_CAL_DRVDN

21.1.4.26 APB_MISC_GP_CAM_RST_CFGPADCTRL_0

CAM_RST_CFG Pad control register

Offset: 0x93c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CAM_RST_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CAM_RST_CFG_CAL_DRVDN

21.1.4.27 APB_MISC_GP_CLK_32K_IN_CFGPADCTRL_0

CLK_32K_IN_CFG Pad control register

Offset: 0x940 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CLK_32K_IN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CLK_32K_IN_CFG_CAL_DRVDN

21.1.4.28 APB_MISC_GP_CLK_32K_OUT_CFGPADCTRL_0

CLK_32K_OUT_CFG Pad control register

Offset: 0x944 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CLK_32K_OUT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CLK_32K_OUT_CFG_CAL_DRVDN

21.1.4.29 APB_MISC_GP_CLK_REQ_CFGPADCTRL_0

CLK_REQ_CFG Pad control register

Offset: 0x948 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CLK_REQ_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CLK_REQ_CFG_CAL_DRVDN

21.1.4.30 APB_MISC_GP_CORE_PWR_REQ_CFGPADCTRL_0

CORE_PWR_REQ_CFG Pad control register

Offset: 0x94c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CORE_PWR_REQ_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CORE_PWR_REQ_CFG_CAL_DRVDN

21.1.4.31 APB_MISC_GP_CPU_PWR_REQ_CFGPADCTRL_0

CPU_PWR_REQ_CFG Pad control register

Offset: 0x950 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_CPU_PWR_REQ_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_CPU_PWR_REQ_CFG_CAL_DRVDN

21.1.4.32 APB_MISC_GP_DAP1_DIN_CFGPADCTRL_0

DAP1_DIN_CFG Pad control register

Offset: 0x954 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP1_DIN_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP1_DIN_CFG_CAL_DRVDN_SLWR

21.1.4.33 APB_MISC_GP_DAP1_DOUT_CFGPADCTRL_0

DAP1_DOUT_CFG Pad control register

Offset: 0x958 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP1_DOUT_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP1_DOUT_CFG_CAL_DRVDN_SLWR

21.1.4.34 APB_MISC_GP_DAP1_FS_CFGPADCTRL_0

DAP1_FS_CFG Pad control register

Offset: 0x95c | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP1_FS_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP1_FS_CFG_CAL_DRVDN_SLWR

21.1.4.35 APB_MISC_GP_DAP1_SCLK_CFGPADCTRL_0

DAP1_SCLK_CFG Pad control register

Offset: 0x960 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP1_SCLK_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP1_SCLK_CFG_CAL_DRVDN_SLWR

21.1.4.36 APB_MISC_GP_DAP2_DIN_CFGPADCTRL_0

DAP2_DIN_CFG Pad control register

Offset: 0x964 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP2_DIN_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP2_DIN_CFG_CAL_DRVDN_SLWR

21.1.4.37 APB_MISC_GP_DAP2_DOUT_CFGPADCTRL_0

DAP2_DOUT_CFG Pad control register

Offset: 0x968 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP2_DOUT_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP2_DOUT_CFG_CAL_DRVDN_SLWR

21.1.4.38 APB_MISC_GP_DAP2_FS_CFGPADCTRL_0

DAP2_FS_CFG Pad control register

Offset: 0x96c | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP2_FS_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP2_FS_CFG_CAL_DRVDN_SLWR

21.1.4.39 APB_MISC_GP_DAP2_SCLK_CFGPADCTRL_0

DAP2_SCLK_CFG Pad control register

Offset: 0x970 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_DAP2_SCLK_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DAP2_SCLK_CFG_CAL_DRVDN_SLWR

21.1.4.40 APB_MISC_GP_DAP4_DIN_CFGPADCTRL_0

DAP4_DIN_CFG Pad control register

Offset: 0x974 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DAP4_DIN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DAP4_DIN_CFG_CAL_DRVDN

21.1.4.41 APB_MISC_GP_DAP4_DOUT_CFGPADCTRL_0

DAP4_DOUT_CFG Pad control register

Offset: 0x978 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DAP4_DOUT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DAP4_DOUT_CFG_CAL_DRVDN

21.1.4.42 APB_MISC_GP_DAP4_FS_CFGPADCTRL_0

DAP4_FS_CFG Pad control register

Offset: 0x97c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DAP4_FS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DAP4_FS_CFG_CAL_DRVDN

21.1.4.43 APB_MISC_GP_DAP4_SCLK_CFGPADCTRL_0

DAP4_SCLK_CFG Pad control register

Offset: 0x980 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DAP4_SCLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DAP4_SCLK_CFG_CAL_DRVDN

21.1.4.44 APB_MISC_GP_DMIC1_CLK_CFGPADCTRL_0

DMIC1_CLK_CFG Pad control register

Offset: 0x984 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DMIC1_CLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DMIC1_CLK_CFG_CAL_DRVDN

21.1.4.45 APB_MISC_GP_DMIC1_DAT_CFGPADCTRL_0

DMIC1_DAT_CFG Pad control register

Offset: 0x988 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DMIC1_DAT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DMIC1_DAT_CFG_CAL_DRVDN

21.1.4.46 APB_MISC_GP_DMIC2_CLK_CFGPADCTRL_0

DMIC2_CLK_CFG Pad control register

Offset: 0x98c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DMIC2_CLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DMIC2_CLK_CFG_CAL_DRVDN

21.1.4.47 APB_MISC_GP_DMIC2_DAT_CFGPADCTRL_0

DMIC2_DAT_CFG Pad control register

Offset: 0x990 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DMIC2_DAT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DMIC2_DAT_CFG_CAL_DRVDN

21.1.4.48 APB_MISC_GP_DMIC3_CLK_CFGPADCTRL_0

DMIC3_CLK_CFG Pad control register

Offset: 0x994 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DMIC3_CLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DMIC3_CLK_CFG_CAL_DRVDN

21.1.4.49 APB_MISC_GP_DMIC3_DAT_CFGPADCTRL_0

DMIC3_DAT_CFG Pad control register

Offset: 0x998 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DMIC3_DAT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DMIC3_DAT_CFG_CAL_DRVDN

21.1.4.50 APB_MISC_GP_DP_HPD_CFGPADCTRL_0

DP_HPD_CFG Pad control register

Offset: 0x99c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DP_HPD0_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DP_HPD0_CFG_CAL_DRVDN

21.1.4.51 APB_MISC_GP_DVFS_CLK_CFGPADCTRL_0

DVFS_CLK_CFG Pad control register

Offset: 0x9a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DVFS_CLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DVFS_CLK_CFG_CAL_DRVDN

21.1.4.52 APB_MISC_GP_DVFS_PWM_CFGPADCTRL_0

DVFS_PWM_CFG Pad control register

Offset: 0x9a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_DVFS_PWM_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DVFS_PWM_CFG_CAL_DRVDN

21.1.4.53 APB_MISC_GP_GEN1_I2C_SCL_CFGPADCTRL_0

GEN1_I2C_SCL_CFG Pad control register

Offset: 0x9a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GEN1_I2C_SCL_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GEN1_I2C_SCL_CFG_CAL_DRVDN

21.1.4.54 APB_MISC_GP_GEN1_I2C_SDA_CFGPADCTRL_0

GEN1_I2C_SDA_CFG Pad control register

Offset: 0x9ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GEN1_I2C_SDA_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GEN1_I2C_SDA_CFG_CAL_DRVDN

21.1.4.55 APB_MISC_GP_GEN2_I2C_SCL_CFGPADCTRL_0

GEN2_I2C_SCL_CFG Pad control register

Offset: 0x9b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GEN2_I2C_SCL_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GEN2_I2C_SCL_CFG_CAL_DRVDN

21.1.4.56 APB_MISC_GP_GEN2_I2C_SDA_CFGPADCTRL_0

GEN2_I2C_SDA_CFG Pad control register

Offset: 0x9b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GEN2_I2C_SDA_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GEN2_I2C_SDA_CFG_CAL_DRVDN

21.1.4.57 APB_MISC_GP_GEN3_I2C_SCL_CFGPADCTRL_0

GEN3_I2C_SCL_CFG Pad control register

Offset: 0x9b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GEN3_I2C_SCL_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GEN3_I2C_SCL_CFG_CAL_DRVDN

21.1.4.58 APB_MISC_GP_GEN3_I2C_SDA_CFGPADCTRL_0

GEN3_I2C_SDA_CFG Pad control register

Offset: 0x9bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GEN3_I2C_SDA_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GEN3_I2C_SDA_CFG_CAL_DRVDN

21.1.4.59 APB_MISC_GP_GPIO_PA6_CFGPADCTRL_0

GPIO_PA6_CFG Pad control register

Offset: 0x9c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPIO_PA6_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPIO_PA6_CFG_CAL_DRVDN

21.1.4.60 APB_MISC_GP_GPIO_PCC7_CFGPADCTRL_0

GPIO_PCC7_CFG Pad control register

Offset: 0x9c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPIO_PCC7_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPIO_PCC7_CFG_CAL_DRVDN

21.1.4.61 APB_MISC_GP_GPIO_PE6_CFGPADCTRL_0

GPIO_PE6_CFG Pad control register

Offset: 0x9c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPIO_PE6_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPIO_PE6_CFG_CAL_DRVDN

21.1.4.62 APB_MISC_GP_GPIO_PE7_CFGPADCTRL_0

GPIO_PE7_CFG Pad control register

Offset: 0x9cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPIO_PE7_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPIO_PE7_CFG_CAL_DRVDN

21.1.4.63 APB_MISC_GP_GPIO_PH6_CFGPADCTRL_0

GPIO_PH6_CFG Pad control register

Offset: 0x9d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPIO_PH6_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPIO_PH6_CFG_CAL_DRVDN

21.1.4.64 APB_MISC_GP_GPIO_PK0_CFGPADCTRL_0

GPIO_PK0_CFG Pad control register

Offset: 0x9d4 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK0_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK0_CFG_CAL_DRVDN_SLWR

21.1.4.65 APB_MISC_GP_GPIO_PK1_CFGPADCTRL_0

GPIO_PK1_CFG Pad control register

Offset: 0x9d8 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK1_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK1_CFG_CAL_DRVDN_SLWR

21.1.4.66 APB_MISC_GP_GPIO_PK2_CFGPADCTRL_0

GPIO_PK2_CFG Pad control register

Offset: 0x9dc | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK2_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK2_CFG_CAL_DRVDN_SLWR

21.1.4.67 APB_MISC_GP_GPIO_PK3_CFGPADCTRL_0

GPIO_PK3_CFG Pad control register

Offset: 0x9e0 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK3_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK3_CFG_CAL_DRVDN_SLWR

21.1.4.68 APB_MISC_GP_GPIO_PK4_CFGPADCTRL_0

GPIO_PK4_CFG Pad control register

Offset: 0x9e4 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK4_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK4_CFG_CAL_DRVDN_SLWR

21.1.4.69 APB_MISC_GP_GPIO_PK5_CFGPADCTRL_0

GPIO_PK5_CFG Pad control register

Offset: 0x9e8 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK5_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK5_CFG_CAL_DRVDN_SLWR

21.1.4.70 APB_MISC_GP_GPIO_PK6_CFGPADCTRL_0

GPIO_PK6_CFG Pad control register

Offset: 0x9ec | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK6_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK6_CFG_CAL_DRVDN_SLWR

21.1.4.71 APB_MISC_GP_GPIO_PK7_CFGPADCTRL_0

GPIO_PK7_CFG Pad control register

Offset: 0x9f0 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PK7_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PK7_CFG_CAL_DRVDN_SLWR

21.1.4.72 APB_MISC_GP_GPIO_PL0_CFGPADCTRL_0

GPIO_PL0_CFG Pad control register

Offset: 0x9f4 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PL0_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PL0_CFG_CAL_DRVDN_SLWR

21.1.4.73 APB_MISC_GP_GPIO_PL1_CFGPADCTRL_0

GPIO_PL1_CFG Pad control register

Offset: 0x9f8 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GPIO_PL1_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_GPIO_PL1_CFG_CAL_DRVDN_SLWR

21.1.4.74 APB_MISC_GP_GPIO_PZ0_CFGPADCTRL_0

GPIO_PZ0_CFG Pad control register

Offset: 0x9fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0000000x0000000xxxxxxxxxxxx)

Bit	Reset	Description
26:20	0x0	CFG2TMC_GPIO_PZ0_CFG_CAL_DRVUP
18:12	0x0	CFG2TMC_GPIO_PZ0_CFG_CAL_DRVDN

21.1.4.75 APB_MISC_GP_GPIO_PZ1_CFGPADCTRL_0

GPIO_PZ1_CFG Pad control register

Offset: 0xa00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0000000x0000000xxxxxxxxxxxx)

Bit	Reset	Description
26:20	0x0	CFG2TMC_GPIO_PZ1_CFG_CAL_DRVUP
18:12	0x0	CFG2TMC_GPIO_PZ1_CFG_CAL_DRVDN

21.1.4.76 APB_MISC_GP_GPIO_PZ2_CFGPADCTRL_0

GPIO_PZ2_CFG Pad control register

Offset: 0xa04 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0000000x0000000xxxxxxxxxxxx)

Bit	Reset	Description
26:20	0x0	CFG2TMC_GPIO_PZ2_CFG_CAL_DRVUP
18:12	0x0	CFG2TMC_GPIO_PZ2_CFG_CAL_DRVDN

21.1.4.77 APB_MISC_GP_GPIO_PZ3_CFGPADCTRL_0

GPIO_PZ3_CFG Pad control register

Offset: 0xa08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0000000x0000000xxxxxxxxxxxx)

Bit	Reset	Description
26:20	0x0	CFG2TMC_GPIO_PZ3_CFG_CAL_DRVUP
18:12	0x0	CFG2TMC_GPIO_PZ3_CFG_CAL_DRVDN

21.1.4.78 APB_MISC_GP_GPIO_PZ4_CFGPADCTRL_0

GPIO_PZ4_CFG Pad control register

Offset: 0xa0c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0000000x0000000xxxxxxxxxxxx)

Bit	Reset	Description
26:20	0x0	CFG2TMC_GPIO_PZ4_CFG_CAL_DRVUP
18:12	0x0	CFG2TMC_GPIO_PZ4_CFG_CAL_DRVDN

21.1.4.79 APB_MISC_GP_GPIO_PZ5_CFGPADCTRL_0

GPIO_PZ5_CFG Pad control register

Offset: 0xa10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0000000x0000000xxxxxxxxxxxx)

Bit	Reset	Description
26:20	0x0	CFG2TMC_GPIO_PZ5_CFG_CAL_DRVUP
18:12	0x0	CFG2TMC_GPIO_PZ5_CFG_CAL_DRVDN

21.1.4.80 APB_MISC_GP_GPIO_X1_AUD_CFGPADCTRL_0

GPIO_X1_AUD_CFG Pad control register

Offset: 0xa14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPIO_X1_AUD_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPIO_X1_AUD_CFG_CAL_DRVDN

21.1.4.81 APB_MISC_GP_GPIO_X3_AUD_CFGPADCTRL_0

GPIO_X3_AUD_CFG Pad control register

Offset: 0xa18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPIO_X3_AUD_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPIO_X3_AUD_CFG_CAL_DRVDN

21.1.4.82 APB_MISC_GP_GPS_EN_CFGPADCTRL_0

GPS_EN_CFG Pad control register

Offset: 0xa1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPS_EN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPS_EN_CFG_CAL_DRVDN

21.1.4.83 APB_MISC_GP_GPS_RST_CFGPADCTRL_0

GPS_RST_CFG Pad control register

Offset: 0xa20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_GPS_RST_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_GPS_RST_CFG_CAL_DRVDN

21.1.4.84 APB_MISC_GP_HDMI_CEC_CFGPADCTRL_0

HDMI_CEC_CFG Pad control register

Offset: 0xa24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_HDMI_CEC_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_HDMI_CEC_CFG_CAL_DRVDN

21.1.4.85 APB_MISC_GP_HDMI_INT_DP_HPD_CFGPADCTRL_0

HDMI_INT_DP_HPD_CFG Pad control register

Offset: 0xa28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_HDMI_INT_DP_HPD_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_HDMI_INT_DP_HPD_CFG_CAL_DRVDN

21.1.4.86 APB_MISC_GP_JTAG_RTCK_CFGPADCTRL_0

JTAG_RTCK_CFG Pad control register

Offset: 0xa2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_JTAG_RTCK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_JTAG_RTCK_CFG_CAL_DRVDN

21.1.4.87 APB_MISC_GP_LCD_BL_EN_CFGPADCTRL_0

LCD_BL_EN_CFG Pad control register

Offset: 0xa30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_LCD_BL_EN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_LCD_BL_EN_CFG_CAL_DRVDN

21.1.4.88 APB_MISC_GP_LCD_BL_PWM_CFGPADCTRL_0

LCD_BL_PWM_CFG Pad control register

Offset: 0xa34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_LCD_BL_PWM_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_LCD_BL_PWM_CFG_CAL_DRVDN

21.1.4.89 APB_MISC_GP_LCD_GPIO1_CFGPADCTRL_0

LCD_GPIO1_CFG Pad control register

Offset: 0xa38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_LCD_GPIO1_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_LCD_GPIO1_CFG_CAL_DRVDN

21.1.4.90 APB_MISC_GP_LCD_GPIO2_CFGPADCTRL_0

LCD_GPIO2_CFG Pad control register

Offset: 0xa3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_LCD_GPIO2_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_LCD_GPIO2_CFG_CAL_DRVDN

21.1.4.91 APB_MISC_GP_LCD_RST_CFGPADCTRL_0

LCD_RST_CFG Pad control register

Offset: 0xa40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_LCD_RST_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_LCD_RST_CFG_CAL_DRVDN

21.1.4.92 APB_MISC_GP_LCD_TE_CFGPADCTRL_0

LCD_TE_CFG Pad control register

Offset: 0xa44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_LCD_TE_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_LCD_TE_CFG_CAL_DRVDN

21.1.4.93 APB_MISC_GP_MODEM_WAKE_AP_CFGPADCTRL_0

MODEM_WAKE_AP_CFG Pad control register

Offset: 0xa48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_MODEM_WAKE_AP_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_MODEM_WAKE_AP_CFG_CAL_DRVDN

21.1.4.94 APB_MISC_GP_MOTION_INT_CFGPADCTRL_0

MOTION_INT_CFG Pad control register

Offset: 0xa4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_MOTION_INT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_MOTION_INT_CFG_CAL_DRVDN

21.1.4.95 APB_MISC_GP_NFC_EN_CFGPADCTRL_0

NFC_EN_CFG Pad control register

Offset: 0xa50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_NFC_EN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_NFC_EN_CFG_CAL_DRVDN

21.1.4.96 APB_MISC_GP_NFC_INT_CFGPADCTRL_0

NFC_INT_CFG Pad control register

Offset: 0xa54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_NFC_INT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_NFC_INT_CFG_CAL_DRVDN

21.1.4.97 APB_MISC_GP_PEX_L0_CLKREQ_N_CFGPADCTRL_0

PEX_L0_CLKREQ_N_CFG Pad control register

Offset: 0xa58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PEX_L0_CLKREQ_N_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PEX_L0_CLKREQ_N_CFG_CAL_DRVDN

21.1.4.98 APB_MISC_GP_PEX_L0_RST_N_CFGPADCTRL_0

PEX_L0_RST_N_CFG Pad control register

Offset: 0xa5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PEX_L0_RST_N_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PEX_L0_RST_N_CFG_CAL_DRVDN

21.1.4.99 APB_MISC_GP_PEX_L1_CLKREQ_N_CFGPADCTRL_0

PEX_L1_CLKREQ_N_CFG Pad control register

Offset: 0xa60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PEX_L1_CLKREQ_N_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PEX_L1_CLKREQ_N_CFG_CAL_DRVDN

21.1.4.100 APB_MISC_GP_PEX_L1_RST_N_CFGPADCTRL_0

PEX_L1_RST_N_CFG Pad control register

Offset: 0xa64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PEX_L1_RST_N_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PEX_L1_RST_N_CFG_CAL_DRVDN

21.1.4.101 APB_MISC_GP_PEX_WAKE_N_CFGPADCTRL_0

PEX_WAKE_N_CFG Pad control register

Offset: 0xa68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PEX_WAKE_N_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PEX_WAKE_N_CFG_CAL_DRVDN

21.1.4.102 APB_MISC_GP_PWR_I2C_SCL_CFGPADCTRL_0

PWR_I2C_SCL_CFG Pad control register

Offset: 0xa6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PWR_I2C_SCL_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PWR_I2C_SCL_CFG_CAL_DRVDN

21.1.4.103 APB_MISC_GP_PWR_I2C_SDA_CFGPADCTRL_0

PWR_I2C_SDA_CFG Pad control register

Offset: 0xa70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PWR_I2C_SDA_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PWR_I2C_SDA_CFG_CAL_DRVDN

21.1.4.104 APB_MISC_GP_PWR_INT_N_CFGPADCTRL_0

PWR_INT_N_CFG Pad control register

Offset: 0xa74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_PWR_INT_N_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_PWR_INT_N_CFG_CAL_DRVDN

21.1.4.105 APB_MISC_GP_QSPI_COMP_CFGPADCTRL_0

QSPI_COMP_CFG Pad control register

Offset: 0xa78 | Read/Write: R/W | Reset: 0x00808000 (0bxxxxx0001000x0001000xxxxxxxxxxx)

Bit	Reset	Description
26:20	0x8	CFG2TMC_QSPI_COMP_CFG_CAL_DRVUP
18:12	0x8	CFG2TMC_QSPI_COMP_CFG_CAL_DRVDN

21.1.4.106 APB_MISC_GP_QSPI_SCK_CFGPADCTRL_0

QSPI_SCK_CFG Pad control register

Offset: 0xa90 | Read/Write: R/W | Reset: 0x50000000 (0b0101xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x1	CFG2TMC_QSPI_SCK_CFG_CAL_DRVUP_SLWF
29:28	0x1	CFG2TMC_QSPI_SCK_CFG_CAL_DRVDN_SLWR

21.1.4.107 APB_MISC_GP_SATA_LED_ACTIVE_CFGPADCTRL_0

SATA_LED_ACTIVE_CFG Pad control register

Offset: 0xa94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_SATA_LED_ACTIVE_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_SATA_LED_ACTIVE_CFG_CAL_DRVDN

21.1.4.108 APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0

SDMMC1_PAD_CFG Pad control register

Controls for BDSDMEM pads of SDMMC1

The following calibration codes need to be used for the pad under default conditions. In General, the calibration pad will provide the code for the pad.

To bypass calibration and provide default code for the required impedance, these values (in decimal) should be used.

On power ON, Vio is 3.3 V only. Software runs calibration, when Vio is switched to 1.8 V. So, 3.3 V DRV codes should be used as default values.

Table 102: DRV Codes

Supply	33 ohms		50 ohms	
	Up	DN	Up	DN
3.3V			8	8
1.8V			3	4

Offset: 0xa98 | Read/Write: R/W | Reset: 0x00808000 (0b0000x0001000x0001000xxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SDMMC1_CLK_CFG_CAL_DRVUP_SLWF:SlewF code override for CLK pad only. If autocal is disabled in the SDMMC1 controller, this register field value will be applied to SLWF input of sdmmc1 CLK pad only.
29:28	0x0	CFG2TMC_SDMMC1_CLK_CFG_CAL_DRVDN_SLWR:SlewR code override for CLK pad only. If autocal is disabled in the SDMMC1 controller, this register field value will be applied to SLWR input of sdmmc1 CLK pad only.
26:20	0x8	CFG2TMC_SDMMC1_PAD_CAL_DRVUP:3.3V 50ohm driver. If autocal is disabled in the SDMMC1 controller, this register field value will be applied to DRVUP input of CLK/DAT/CMD pads in sdmmc1 pin group
18:12	0x8	CFG2TMC_SDMMC1_PAD_CAL_DRVDN:3.3V 50ohm driver If autocal is disabled in the SDMMC1 controller, this register field value will be applied to DRVDN input of CLK/DAT/CMD pads in sdmmc1 pin group

21.1.4.109 APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0

EMMC2_PAD_E_CFG Pad control register

Offset: 0xa9c | Read/Write: R/W | Reset: 0x07ffc310 (0b0000x11111111111100001100010000)

Bit	Reset	Description
31:30	0x0	CFG2TMC_EMMC2_PAD_DRVUP_SLWF: SLWF code for CLK pad of emmc2 iobrick. If autocal is disabled in the SDMMC2 controller, this register field value is applied to DRVUP_SLWF input of emmc2 iobrick which is sent to CLK pad only.
29:28	0x0	CFG2TMC_EMMC2_PAD_DRVDN_SLWR: SLWR code for CLK pad of emmc2 iobrick. If autocal is disabled in the SDMMC2 controller, this register field value is applied to DRVDN_SLWR input of emmc2 iobrick which is sent to CLK pad only.
26	0x1	MISC2PMC_EMMC2_DAT7_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
25	0x1	MISC2PMC_EMMC2_DAT6_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
24	0x1	MISC2PMC_EMMC2_DAT5_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
23	0x1	MISC2PMC_EMMC2_DAT4_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
22	0x1	MISC2PMC_EMMC2_DAT3_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
21	0x1	MISC2PMC_EMMC2_DAT2_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.

Bit	Reset	Description
20	0x1	MISC2PMC_EMMC2_DAT1_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
19	0x1	MISC2PMC_EMMC2_DAT0_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
18	0x1	MISC2PMC_EMMC2_CMD_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
17	0x1	MISC2PMC_EMMC2_DQSB_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
16	0x1	MISC2PMC_EMMC2_DQS_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
15	0x1	MISC2PMC_EMMC2_CLKB_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
14	0x1	MISC2PMC_EMMC2_CLK_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
13:8	0x3	CFG2TMC_EMMC2_PAD_DRVUP_COMP: DRVUP code for CLK/DAT/CMD pads in emmc2 iobrick. Default: 1.8V 50ohm driver. If autocal is disabled in the SDMMC2 controller, this register field value will be applied to DRVUP_COMP input of emmc2 iobrick which will be level shifted internally and sent to CLK/DAT/CMD pads of iobrick
7:2	0x4	CFG2TMC_EMMC2_PAD_DRVDN_COMP: DRVDN code for CLK/DAT/CMD pads in emmc2 iobrick. Default: 1.8V 50ohm driver. If autocal is disabled in the SDMMC2 controller, this register field value will be applied to DRVDN_COMP input of emmc2 iobrick which will be level shifted internally and sent to CLK/DAT/CMD pads of iobrick
1	0x0	CFG2TMC_EMMC2_PAD_E_PREEMP:1: enables pre-emphasis circuit in emmc2 iobrick
0	0x0	CFG2TMC_EMMC2_PAD_E_SCH:1: enables Schmitt trigger in emmc2 iobrick

21.1.4.110 APB_MISC_GP_EMMC2_PAD_DRV_TYPE_CFGPADCTRL_0

EMMC2_PAD_DRV_TYPE_CFG Pad control register

Selects driver type for each pad in emmc2 iobrick.

- DRV_TYPE[0]: 0 - 66/100 ohm driver; 1 - 33/50 ohm driver
- DRV_TYPE[1]: should be zero always - not used

Offset: 0xaa0 | Read/Write: R/W | Reset: 0x00155555 (0bxxxxxxxxxx01010101010101010101)

Bit	Reset	Description
21:20	0x1	CFG2TMC_EMMC2_PAD_D7_DRV_TYPE: selects driver type for DAT7 pad
19:18	0x1	CFG2TMC_EMMC2_PAD_D6_DRV_TYPE: selects driver type for DAT6 pad
17:16	0x1	CFG2TMC_EMMC2_PAD_D5_DRV_TYPE: selects driver type for DAT5 pad
15:14	0x1	CFG2TMC_EMMC2_PAD_D4_DRV_TYPE: selects driver type for DAT4 pad
13:12	0x1	CFG2TMC_EMMC2_PAD_D3_DRV_TYPE: selects driver type for DAT3 pad
11:10	0x1	CFG2TMC_EMMC2_PAD_D2_DRV_TYPE: selects driver type for DAT2 pad
9:8	0x1	CFG2TMC_EMMC2_PAD_D1_DRV_TYPE: selects driver type for DAT1 pad
7:6	0x1	CFG2TMC_EMMC2_PAD_D0_DRV_TYPE: selects driver type for DAT0 pad
5:4	0x1	CFG2TMC_EMMC2_PAD_CLKB_DRV_TYPE: selects driver type for CLKB pad
3:2	0x1	CFG2TMC_EMMC2_PAD_CLK_DRV_TYPE: selects driver type for CLK pad
1:0	0x1	CFG2TMC_EMMC2_PAD_CMD_DRV_TYPE: selects driver type for CMD pad

21.1.4.111 APB_MISC_GP_EMMC2_PAD_PUPD_CFGPADCTRL_0

EMMC2_PAD_PUPD_CFG Pad control register

Offset: 0xaa4 | Read/Write: R/W | Reset: 0x026aaaa6 (0bxxxxxx10011010101010101010100110)

Bit	Reset	Description
25	0x1	CFG2TMC_EMMC2_PAD_DQSB_PUPD_PULLU

Bit	Reset	Description
24	0x0	CFG2TMC_EMMC2_PAD_DQSB_PUPD_PULLD
23	0x0	CFG2TMC_EMMC2_PAD_DQS_PUPD_PULLU
22	0x1	CFG2TMC_EMMC2_PAD_DQS_PUPD_PULLD
21	0x1	CFG2TMC_EMMC2_PAD_D7_PUPD_PULLU
20	0x0	CFG2TMC_EMMC2_PAD_D7_PUPD_PULLD
19	0x1	CFG2TMC_EMMC2_PAD_D6_PUPD_PULLU
18	0x0	CFG2TMC_EMMC2_PAD_D6_PUPD_PULLD
17	0x1	CFG2TMC_EMMC2_PAD_D5_PUPD_PULLU
16	0x0	CFG2TMC_EMMC2_PAD_D5_PUPD_PULLD
15	0x1	CFG2TMC_EMMC2_PAD_D4_PUPD_PULLU
14	0x0	CFG2TMC_EMMC2_PAD_D4_PUPD_PULLD
13	0x1	CFG2TMC_EMMC2_PAD_D3_PUPD_PULLU
12	0x0	CFG2TMC_EMMC2_PAD_D3_PUPD_PULLD
11	0x1	CFG2TMC_EMMC2_PAD_D2_PUPD_PULLU
10	0x0	CFG2TMC_EMMC2_PAD_D2_PUPD_PULLD
9	0x1	CFG2TMC_EMMC2_PAD_D1_PUPD_PULLU
8	0x0	CFG2TMC_EMMC2_PAD_D1_PUPD_PULLD
7	0x1	CFG2TMC_EMMC2_PAD_D0_PUPD_PULLU
6	0x0	CFG2TMC_EMMC2_PAD_D0_PUPD_PULLD
5	0x1	CFG2TMC_EMMC2_PAD_CLKB_PUPD_PULLU
4	0x0	CFG2TMC_EMMC2_PAD_CLKB_PUPD_PULLD
3	0x0	CFG2TMC_EMMC2_PAD_CLK_PUPD_PULLU
2	0x1	CFG2TMC_EMMC2_PAD_CLK_PUPD_PULLD
1	0x1	CFG2TMC_EMMC2_PAD_CMD_PUPD_PULLU
0	0x0	CFG2TMC_EMMC2_PAD_CMD_PUPD_PULLD: PULLD: enables weak pull down; PULLDU: enables weak pull up.

21.1.4.112 APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0

SDMMC3_PAD_CFG Pad control register

Controls for BDSDMEM pads of SDMMC3.

The following calibration codes need to be used for the pad under default conditions. In general, the calibration pad provides the code for the pad.

To bypass calibration and provide default code for the required impedance, these values (in decimal) should be used.

On power ON, Vio is 3.3 V only. Software runs calibration, when Vio is switched to 1.8 V. So, we should use 3.3 V DRV codes as default values

Table 103: DRV Codes

Supply	33 ohms		50 ohms	
	Up	DN	Up	DN
3.3V			8	8
1.8V			3	4

Offset: 0xab0 | Read/Write: R/W | Reset: 0x00808000 (0b0000x0001000x0001000xxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SDMMC3_CLK_CFG_CAL_DRVUP_SLWF: SlewF code override for CLK pad only. If autocal is disabled in the SDMMC3 controller, this register field value will be applied to SLWF input of sdmmc3 CLK pad only.

Bit	Reset	Description
29:28	0x0	CFG2TMC_SDMMC3_CLK_CFG_CAL_DRVDN_SLWR: SlewR code override for CLK pad only. If autocal is disabled in the SDMMC3 controller, this register field value will be applied to SLWR input of sdmmc3 CLK pad only.
26:20	0x8	CFG2TMC_SDMMC3_PAD_CAL_DRVUP:3.3V 50ohm driver. If autocal is disabled in the SDMMC3 controller, this register field value will be applied to DRVUP input of CLK/DAT/CMD pads in sdmmc3 pin group
18:12	0x8	CFG2TMC_SDMMC3_PAD_CAL_DRVDN:3.3V 50ohm driver If autocal is disabled in the SDMMC3 controller, this register field value will be applied to DRVDN input of CLK/DAT/CMD pads in sdmmc3 pin group

21.1.4.113 APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0

EMMC4_PAD_E_CFG Pad control register

Offset: 0xab4 | Read/Write: R/W | Reset: 0x07ffc310 (0b0000x11111111111100001100010000)

Bit	Reset	Description
31:30	0x0	CFG2TMC_EMMC4_PAD_DRVUP_SLWF: SLWF code for CLK pad of emmc4 iobrick. If autocal is disabled in the SDMMC4 controller, this register field value will be applied to DRVUP_SLWF input of emmc4 iobrick which will be sent to CLK pad only.
29:28	0x0	CFG2TMC_EMMC4_PAD_DRVDN_SLWR: SLWR code for CLK pad of emmc4 iobrick. If autocal is disabled in the SDMMC4 controller, this register field value will be applied to DRVDN_SLWR input of emmc4 iobrick which will be sent to CLK pad only.
26	0x1	MISC2PMC_EMMC4_DAT7_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
25	0x1	MISC2PMC_EMMC4_DAT6_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
24	0x1	MISC2PMC_EMMC4_DAT5_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
23	0x1	MISC2PMC_EMMC4_DAT4_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
22	0x1	MISC2PMC_EMMC4_DAT3_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
21	0x1	MISC2PMC_EMMC4_DAT2_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
20	0x1	MISC2PMC_EMMC4_DAT1_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
19	0x1	MISC2PMC_EMMC4_DAT0_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
18	0x1	MISC2PMC_EMMC4_CMD_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
17	0x1	MISC2PMC_EMMC4_DQSB_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
16	0x1	MISC2PMC_EMMC4_DQS_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
15	0x1	MISC2PMC_EMMC4_CLKB_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
14	0x1	MISC2PMC_EMMC4_CLK_PARK:0: Normal - 1: PARKED: default value should be 1 so that during LP0 exit the pads stay in DPD mode until pinmux recovery is done and I/O controllers are activated, etc.
13:8	0x3	CFG2TMC_EMMC4_PAD_DRVUP_COMP: DRVUP code for CLK/DAT/CMD pads in emmc4 iobrick. Default: 1.8V 50ohm driver. If autocal is disabled in the SDMMC4 controller, this register field value will be applied to DRVUP_COMP input of emmc4 iobrick which will be level shifted internally and sent to CLK/DAT/CMD pads of iobrick
7:2	0x4	CFG2TMC_EMMC4_PAD_DRVDN_COMP: DRVDN code for CLK/DAT/CMD pads in emmc4 iobrick. Default: 1.8V 50ohm driver. If autocal is disabled in the SDMMC4 controller, this register field value will be applied to DRVDN_COMP input of emmc4 iobrick which will be level shifted internally and sent to CLK/DAT/CMD pads of iobrick
1	0x0	CFG2TMC_EMMC4_PAD_E_PREEMP:1:enables pre-emphasis circuit in emmc4 iobrick
0	0x0	CFG2TMC_EMMC4_PAD_E_SCH:1: enables Schmitt trigger in emmc4 iobrick

21.1.4.114 APB_MISC_GP_EMMC4_PAD_DRV_TYPE_CFGPADCTRL_0

EMMC4_PAD_DRV_TYPE_CFG Pad control register

Selects driver type for each pad in emmc4 iobrick.

- DRV_TYPE[0] : 0 - 66/100 ohm driver; 1 - 33/50 ohm driver
- DRV_TYPE[1] : should be zero always - not used

Offset: 0xab8 | Read/Write: R/W | Reset: 0x00155555 (0bxxxxxxxxxxx01010101010101010101)

Bit	Reset	Description
21:20	0x1	CFG2TMC_EMMC4_PAD_D7_DRV_TYPE: selects driver type for DAT7 pad
19:18	0x1	CFG2TMC_EMMC4_PAD_D6_DRV_TYPE: selects driver type for DAT6 pad
17:16	0x1	CFG2TMC_EMMC4_PAD_D5_DRV_TYPE: selects driver type for DAT5 pad
15:14	0x1	CFG2TMC_EMMC4_PAD_D4_DRV_TYPE: selects driver type for DAT4 pad
13:12	0x1	CFG2TMC_EMMC4_PAD_D3_DRV_TYPE: selects driver type for DAT3 pad
11:10	0x1	CFG2TMC_EMMC4_PAD_D2_DRV_TYPE: selects driver type for DAT2 pad
9:8	0x1	CFG2TMC_EMMC4_PAD_D1_DRV_TYPE: selects driver type for DAT1 pad
7:6	0x1	CFG2TMC_EMMC4_PAD_D0_DRV_TYPE: selects driver type for DAT0 pad
5:4	0x1	CFG2TMC_EMMC4_PAD_CLKB_DRV_TYPE: selects driver type for CLKB pad
3:2	0x1	CFG2TMC_EMMC4_PAD_CLK_DRV_TYPE: selects driver type for CLK pad
1:0	0x1	CFG2TMC_EMMC4_PAD_CMD_DRV_TYPE: selects driver type for CMD pad

21.1.4.115 APB_MISC_GP_EMMC4_PAD_PUPD_CFGPADCTRL_0

EMMC4_PAD_PUPD_CFG Pad control register

Offset: 0xabc | Read/Write: R/W | Reset: 0x026aaaa6 (0bxxxxxx10011010101010101010100110)

Bit	Reset	Description
25	0x1	CFG2TMC_EMMC4_PAD_DQSB_PUPD_PULLU
24	0x0	CFG2TMC_EMMC4_PAD_DQSB_PUPD_PULLD
23	0x0	CFG2TMC_EMMC4_PAD_DQS_PUPD_PULLU
22	0x1	CFG2TMC_EMMC4_PAD_DQS_PUPD_PULLD
21	0x1	CFG2TMC_EMMC4_PAD_D7_PUPD_PULLU
20	0x0	CFG2TMC_EMMC4_PAD_D7_PUPD_PULLD
19	0x1	CFG2TMC_EMMC4_PAD_D6_PUPD_PULLU
18	0x0	CFG2TMC_EMMC4_PAD_D6_PUPD_PULLD
17	0x1	CFG2TMC_EMMC4_PAD_D5_PUPD_PULLU
16	0x0	CFG2TMC_EMMC4_PAD_D5_PUPD_PULLD
15	0x1	CFG2TMC_EMMC4_PAD_D4_PUPD_PULLU
14	0x0	CFG2TMC_EMMC4_PAD_D4_PUPD_PULLD
13	0x1	CFG2TMC_EMMC4_PAD_D3_PUPD_PULLU
12	0x0	CFG2TMC_EMMC4_PAD_D3_PUPD_PULLD
11	0x1	CFG2TMC_EMMC4_PAD_D2_PUPD_PULLU
10	0x0	CFG2TMC_EMMC4_PAD_D2_PUPD_PULLD
9	0x1	CFG2TMC_EMMC4_PAD_D1_PUPD_PULLU
8	0x0	CFG2TMC_EMMC4_PAD_D1_PUPD_PULLD
7	0x1	CFG2TMC_EMMC4_PAD_D0_PUPD_PULLU
6	0x0	CFG2TMC_EMMC4_PAD_D0_PUPD_PULLD
5	0x1	CFG2TMC_EMMC4_PAD_CLKB_PUPD_PULLU
4	0x0	CFG2TMC_EMMC4_PAD_CLKB_PUPD_PULLD

Bit	Reset	Description
3	0x0	CFG2TMC_EMMC4_PAD_CLK_PUPD_PULLU
2	0x1	CFG2TMC_EMMC4_PAD_CLK_PUPD_PULLD
1	0x1	CFG2TMC_EMMC4_PAD_CMD_PUPD_PULLU
0	0x0	CFG2TMC_EMMC4_PAD_CMD_PUPD_PULLD: PULLD: enables weak pull down; PULLDU: enables weak pull up.

21.1.4.116 APB_MISC_GP_SHUTDOWN_CFGPADCTRL_0

SHUTDOWN_CFG Pad control register

Offset: 0xac8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_SHUTDOWN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_SHUTDOWN_CFG_CAL_DRVDN

21.1.4.117 APB_MISC_GP_SPDIF_IN_CFGPADCTRL_0

SPDIF_IN_CFG Pad control register

Offset: 0xacc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_SPDIF_IN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_SPDIF_IN_CFG_CAL_DRVDN

21.1.4.118 APB_MISC_GP_SPDIF_OUT_CFGPADCTRL_0

SPDIF_OUT_CFG Pad control register

Offset: 0xad0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_SPDIF_OUT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_SPDIF_OUT_CFG_CAL_DRVDN

21.1.4.119 APB_MISC_GP_SPI1_CS0_CFGPADCTRL_0

SPI1_CS0_CFG Pad control register

Offset: 0xad4 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI1_CS0_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI1_CS0_CFG_CAL_DRVDN_SLWR

21.1.4.120 APB_MISC_GP_SPI1_CS1_CFGPADCTRL_0

SPI1_CS1_CFG Pad control register

Offset: 0xad8 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI1_CS1_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI1_CS1_CFG_CAL_DRVDN_SLWR

21.1.4.121 APB_MISC_GP_SPI1_MISO_CFGPADCTRL_0

SPI1_MISO_CFG Pad control register

Offset: 0xadc | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI1_MISO_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI1_MISO_CFG_CAL_DRVDN_SLWR

21.1.4.122 APB_MISC_GP_SPI1_MOSI_CFGPADCTRL_0

SPI1_MOSI_CFG Pad control register

Offset: 0xae0 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI1_MOSI_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI1_MOSI_CFG_CAL_DRVDN_SLWR

21.1.4.123 APB_MISC_GP_SPI1_SCK_CFGPADCTRL_0

SPI1_SCK_CFG Pad control register

Offset: 0xae4 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI1_SCK_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI1_SCK_CFG_CAL_DRVDN_SLWR

21.1.4.124 APB_MISC_GP_SPI2_CS0_CFGPADCTRL_0

SPI2_CS0_CFG Pad control register

Offset: 0xae8 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI2_CS0_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI2_CS0_CFG_CAL_DRVDN_SLWR

21.1.4.125 APB_MISC_GP_SPI2_CS1_CFGPADCTRL_0

SPI2_CS1_CFG Pad control register

Offset: 0xaec | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI2_CS1_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI2_CS1_CFG_CAL_DRVDN_SLWR

21.1.4.126 APB_MISC_GP_SPI2_MISO_CFGPADCTRL_0

SPI2_MISO_CFG Pad control register

Offset: 0xaf0 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI2_MISO_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI2_MISO_CFG_CAL_DRVDN_SLWR

21.1.4.127 APB_MISC_GP_SPI2_MOSI_CFGPADCTRL_0

SPI2_MOSI_CFG Pad control register

Offset: 0xaf4 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI2_MOSI_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI2_MOSI_CFG_CAL_DRVDN_SLWR

21.1.4.128 APB_MISC_GP_SPI2_SCK_CFGPADCTRL_0

SPI2_SCK_CFG Pad control register

Offset: 0xaf8 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI2_SCK_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI2_SCK_CFG_CAL_DRVDN_SLWR

21.1.4.129 APB_MISC_GP_SPI4_CS0_CFGPADCTRL_0

SPI4_CS0_CFG Pad control register

Offset: 0xafc | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI4_CS0_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI4_CS0_CFG_CAL_DRVDN_SLWR

21.1.4.130 APB_MISC_GP_SPI4_MISO_CFGPADCTRL_0

SPI4_MISO_CFG Pad control register

Offset: 0xb00 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI4_MISO_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI4_MISO_CFG_CAL_DRVDN_SLWR

21.1.4.131 APB_MISC_GP_SPI4_MOSI_CFGPADCTRL_0

SPI4_MOSI_CFG Pad control register

Offset: 0xb04 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI4_MOSI_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI4_MOSI_CFG_CAL_DRVDN_SLWR

21.1.4.132 APB_MISC_GP_SPI4_SCK_CFGPADCTRL_0

SPI4_SCK_CFG Pad control register

Offset: 0xb08 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SPI4_SCK_CFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_SPI4_SCK_CFG_CAL_DRVDN_SLWR

21.1.4.133 APB_MISC_GP_TEMP_ALERT_CFGPADCTRL_0

TEMP_ALERT_CFG Pad control register

Offset: 0xb0c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_TEMP_ALERT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_TEMP_ALERT_CFG_CAL_DRVDN

21.1.4.134 APB_MISC_GP_TOUCH_CLK_CFGPADCTRL_0

TOUCH_CLK_CFG Pad control register

Offset: 0xb10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_TOUCH_CLK_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_TOUCH_CLK_CFG_CAL_DRVDN

21.1.4.135 APB_MISC_GP_TOUCH_INT_CFGPADCTRL_0

TOUCH_INT_CFG Pad control register

Offset: 0xb14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_TOUCH_INT_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_TOUCH_INT_CFG_CAL_DRVDN

21.1.4.136 APB_MISC_GP_TOUCH_RST_CFGPADCTRL_0

TOUCH_RST_CFG Pad control register

Offset: 0xb18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_TOUCH_RST_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_TOUCH_RST_CFG_CAL_DRVDN

21.1.4.137 APB_MISC_GP_UART1_CTS_CFGPADCTRL_0

UART1_CTS_CFG Pad control register

Offset: 0xb1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART1_CTS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART1_CTS_CFG_CAL_DRVDN

21.1.4.138 APB_MISC_GP_UART1_RTS_CFGPADCTRL_0

UART1_RTS_CFG Pad control register

Offset: 0xb20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART1_RTS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART1_RTS_CFG_CAL_DRVDN

21.1.4.139 APB_MISC_GP_UART1_RX_CFGPADCTRL_0

UART1_RX_CFG Pad control register

Offset: 0xb24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART1_RX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART1_RX_CFG_CAL_DRVDN

21.1.4.140 APB_MISC_GP_UART1_TX_CFGPADCTRL_0

UART1_TX_CFG Pad control register

Offset: 0xb28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART1_TX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART1_TX_CFG_CAL_DRVDN

21.1.4.141 APB_MISC_GP_UART2_CTS_CFGPADCTRL_0

UART2_CTS_CFG Pad control register

Offset: 0xb2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART2_CTS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART2_CTS_CFG_CAL_DRVDN

21.1.4.142 APB_MISC_GP_UART2_RTS_CFGPADCTRL_0

UART2_RTS_CFG Pad control register

Offset: 0xb30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART2_RTS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART2_RTS_CFG_CAL_DRVDN

21.1.4.143 APB_MISC_GP_UART2_RX_CFGPADCTRL_0

UART2_RX_CFG Pad control register

Offset: 0xb34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART2_RX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART2_RX_CFG_CAL_DRVDN

21.1.4.144 APB_MISC_GP_UART2_TX_CFGPADCTRL_0

UART2_TX_CFG Pad control register

Offset: 0xb38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART2_TX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART2_TX_CFG_CAL_DRVDN

21.1.4.145 APB_MISC_GP_UART3_CTS_CFGPADCTRL_0

UART3_CTS_CFG Pad control register

Offset: 0xb3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART3_CTS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART3_CTS_CFG_CAL_DRVDN

21.1.4.146 APB_MISC_GP_UART3_RTS_CFGPADCTRL_0

UART3_RTS_CFG Pad control register

Offset: 0xb40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	**0x0**	**CFG2TMC_UART3_RTS_CFG_CAL_DRVUP**
16:12	0x0	CFG2TMC_UART3_RTS_CFG_CAL_DRVDN

21.1.4.147 APB_MISC_GP_UART3_RX_CFGPADCTRL_0

UART3_RX_CFG Pad control register

Offset: 0xb44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART3_RX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART3_RX_CFG_CAL_DRVDN

21.1.4.148 APB_MISC_GP_UART3_TX_CFGPADCTRL_0

UART3_TX_CFG Pad control register

Offset: 0xb48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART3_TX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART3_TX_CFG_CAL_DRVDN

21.1.4.149 APB_MISC_GP_UART4_CTS_CFGPADCTRL_0

UART4_CTS_CFG Pad control register

Offset: 0xb4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART4_CTS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART4_CTS_CFG_CAL_DRVDN

21.1.4.150 APB_MISC_GP_UART4_RTS_CFGPADCTRL_0

UART4_RTS_CFG Pad control register

Offset: 0xb50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART4_RTS_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART4_RTS_CFG_CAL_DRVDN

21.1.4.151 APB_MISC_GP_UART4_RX_CFGPADCTRL_0

UART4_RX_CFG Pad control register

Offset: 0xb54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART4_RX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART4_RX_CFG_CAL_DRVDN

21.1.4.152 APB_MISC_GP_UART4_TX_CFGPADCTRL_0

UART4_TX_CFG Pad control register

Offset: 0xb58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_UART4_TX_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_UART4_TX_CFG_CAL_DRVDN

21.1.4.153 APB_MISC_GP_USB_VBUS_EN0_CFGPADCTRL_0

USB_VBUS_EN0_CFG Pad control register

Offset: 0xb5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_USB_VBUS_EN0_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_USB_VBUS_EN0_CFG_CAL_DRVDN

21.1.4.154 APB_MISC_GP_USB_VBUS_EN1_CFGPADCTRL_0

USB_VBUS_EN1_CFG Pad control register

Offset: 0xb60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_USB_VBUS_EN1_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_USB_VBUS_EN1_CFG_CAL_DRVDN

21.1.4.155 APB_MISC_GP_WIFI_EN_CFGPADCTRL_0

WIFI_EN_CFG Pad control register

Offset: 0xb64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_WIFI_EN_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_WIFI_EN_CFG_CAL_DRVDN

21.1.4.156 APB_MISC_GP_WIFI_RST_CFGPADCTRL_0

WIFI_RST_CFG Pad control register

Offset: 0xb68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_WIFI_RST_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_WIFI_RST_CFG_CAL_DRVDN

21.1.4.157 APB_MISC_GP_WIFI_WAKE_AP_CFGPADCTRL_0

WIFI_WAKE_AP_CFG Pad control register

Offset: 0xb6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000xxx00000xxxxxxxxxxx)

Bit	Reset	Description
24:20	0x0	CFG2TMC_WIFI_WAKE_AP_CFG_CAL_DRVUP
16:12	0x0	CFG2TMC_WIFI_WAKE_AP_CFG_CAL_DRVDN

21.1.4.158 APB_MISC_GP_QSPI_COMP_CONTROL_0

QSPI registers

Offset: 0xb70 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000001x)

Bit	R/W	Reset	Description
7	RW	0x0	QSPI_COMP_PAD_REG_ON
6:3	RW	0x0	QSPI_COMP_PAD_VREF_SEL
2	RW	0x0	QSPI_COMP_PAD_CLK
1	RW	0x1	QSPI_COMP_PAD_E_INPUT
0	RO	X	QSPI_COMP_CALIB_STATUS

21.1.4.159 APB_MISC_GP_VGPIO_GPIO_MUX_SEL_0

Offset: 0xb74 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	0x1	SDMMC3_WP_SOURCE: Selects between GPIO and VGPIO for the signal going to SDMMC 0 = GPIO 1 = VGPIO
2	0x1	SDMMC3_CD_SOURCE: Selects between GPIO and VGPIO for the signal going to SDMMC 0 = GPIO 1 = VGPIO
1	0x1	SDMMC1_WP_SOURCE: Selects between GPIO and VGPIO for the signal going to SDMMC 0 = GPIO 1 = VGPIO
0	0x1	SDMMC1_CD_SOURCE: Selects between GPIO and VGPIO for the signal going to SDMMC 0 = GPIO 1 = VGPIO

21.1.4.160 APB_MISC_GP_QSPI_SCK_LPBK_CONTROL_0

Offset: 0xb78 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	QSPI_SCK_PAD_E_LPBK: When set, enables deep loopback in the QSPI pad

21.1.5 SATA Aux Registers

21.1.5.1 SATA_AUX_MISC_CNTL_1_0

MISC_CNTL_1 Register

Offset: 0x1108 | Read/Write: R/W | Reset: 0x00016XX0 (0bxxxxxxxxxxxxxx0010110xxx00xx00000)

Bit	R/W	Reset	Description
18	RW	0x0	AUX_RX_IDLE_STATUS_MASK: Set this bit to mask the aux_rx_idle_status input to the SATA core to 0. 0 = Do not mask the rx_stat_idle input to the SATA core 1 = Mask the rx_stat_idle input to the SATA core to 0. 0 = DISABLE 1 = ENABLE
17	RW	0x0	DEVSLP_OVERRIDE: Set this bit to override the DEVSLP output from the SATA core. 0 = Assertion of DEVSLP is controlled by the SATA core 1 = DEVSLP is asserted irrespective of the state of DEVSLP from the SATA core. 0 = DISABLE 1 = ENABLE
16	RW	0x1	DSP_SUPPORT: Set this bit to set the AHCI's PxDEVSLP.DSP register bit. 0 = Indicates the SATA port does not support device sleep 1 = Indicates the SATA port supports device sleep. 0 = CLEAR 1 = SET
15	RW	0x0	DESO_SUPPORT: Set this bit to set the AHCI's CAP2.DESO register bit. 0 = Indicates the SATA core has the capability to enter DevSleep from any link state 1 = Indicates the SATA core only has the capability to enter DevSleep from the slumber link state. 0 = CLEAR 1 = SET
14	RW	0x1	SADM_SUPPORT: Set this bit to set the AHCI's CAP2.SADM register bit. 0 = Indicates the SATA core does not have the capability to support hardware assertion of the DEVSLP 1 = Indicates the SATA core has the capability to support hardware assertion of the DEVSLP. 0 = CLEAR 1 = SET
13	RW	0x1	SDS_SUPPORT: Set this bit to set the AHCI's CAP2.SAS register bit. 0 = Indicates the SATA core does not support device sleep 1 = Indicates the SATA core supports device sleep. 0 = CLEAR 1 = SET
12	RW	0x0	RX_STAT_IDLE_MASK: Set this bit to mask the rx_stat_idle input to the SATA core to 0. 0 = Do not mask the rx_stat_idle input to the SATA core 1 = Mask the rx_stat_idle input to the SATA core to 0. 0 = DISABLE 1 = ENABLE
11	RO	X	SATA2IPSM_DEVSLP: Indicates if the SATA link is in DEVSLP. Used for debug purposes.
10:9	RO	X	SATA2IPSM_ST: Indicates whether the SATA link is in partial/slumber modes. Used for debug purposes.
8	RW	0x0	NVA2SATA_OOB_ON_SCONTROL_SPD_WR: If this bit is set to 1, the SATA controller will do an OOB and go through speed negotiation with the drive whenever its SCONTROL_SPD register is written. 0 = NO 1 = YES
7	RW	0x0	NVA2SATA_OOB_ON_POR: Controls whether or not SATA controllers do an OOB sequence automatically when they come out of reset. 0 = NO 1 = YES
6:5	RO	X	L0_RX_IDLE_T_SAX: l0_rx_idle_t value from the SATA controller When the SAX partition is power-gated, this field shows a clamped value. 0 = NORMAL

Bit	R/W	Reset	Description
4:3	RW	0x0	L0_RX_IDLE_T_NPG: Sets the l0_rx_idle_t value for the SATA PHY from apb_misc (non-power-gated logic). Before the SAX partition is power-gated, SATA_l0_rx_idle_t_npg needs to have the same value as SATA_l0_rx_idle_t_sax, then set SATA_l0_rx_idle_t_mux to '1'. This ensures that the SATA PHY is still receiving the correct threshold setting for OOB detection when the SAX is power-gated. After SAX power is restored, the same value is restored to the l0_rx_idle_t register in the SATA controller, then SATA_l0_rx_idle_t_mux can be set back to '0'. 0 = NORMAL
2	RW	0x0	L0_RX_IDLE_T_MUX: Select l0_rx_idle_t driving source for SATA PHY 0: From the SATA controller 1: From apb_misc 0 = FROM_SATA 1 = FROM_APB_MISC
1	RW	0x0	PMU2SATA_ACCLMTR_TRIG: This config bit is used inside SATA to select between the two accelerometer triggers, between the external trigger and the PMU's trigger. 0: External trigger disable 1: External trigger enable 0 = DISABLE 1 = ENABLE
0	RW	0x0	DEVICE_DIS_SATA0: Serial ATA Interface 0 Disable 1 = Internal Serial ATA Interface 0 disabled (Not seen as part of PCI space) 0 = Internal Serial ATA Interface 0 enabled. 0 = ENABLE 1 = DISABLE

21.1.5.2 SATA_AUX_RX_STAT_INT_0

RX_STAT_INT Register

Offset: 0x110c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxx0xx0xx0xx)

Bit	R/W	Reset	Description
8	RW	0x0	SATA_DEVSLP_INT_DISABLE: SATA devslp interrupt disable 1 = Disables interrupt assertion when devslp interrupt status is set 0 = Enables interrupt assertion when devslp interrupt status is set 0 = ENABLE 1 = DISABLE
7	RO	X	SATA_DEVSLP: SATA devslp state interrupt status 1 = SATA link is in the devslp state 0 = SATA link is not in the devslp state 0 = NO 1 = YES
6	RO	X	SATA_DEVSLP_INT_STATUS: SATA devslp state interrupt status 1 = devslp interrupt is pending 0 = devslp interrupt is not pending 0 = NONE 1 = PENDING
5	RW	0x0	SATA_DEV_ATTEN_INT_DISABLE: SATA device attention interrupt disable 1 = Disable interrupt assertion when dev_atten interrupt status is set 0 = Enable interrupt assertion when dev_atten interrupt status is set 0 = ENABLE 1 = DISABLE
4	RO	X	SATA_DEVICE_ATTENTION: SATA device attention status. 1: device attention input is asserted 0: device attention input is cleared. 0 = NO 1 = YES
3	RO	X	SATA_DEV_ATTEN_INT_STATUS: SATA device attention interrupt status from GPIO. 1 = Device attention interrupt is pending 0 = Device attention interrupt is not pending 0 = NONE 1 = PENDING

Bit	R/W	Reset	Description
2	RW	0x0	SATA_RX_STAT_INT_DISABLE: SATA rx_stat interrupt disable 1 = Disables interrupt assertion when rx_stat interrupt status is set 0 = Enables interrupt assertion when rx_stat interrupt status is set 0 = ENABLE 1 = DISABLE
1	RO	X	SATA_L0_RX_STAT_IDLE: SATA pad L0 rx_stat_idle status 1: Rx path is idle 0: Rx path is active 0 = NO 1 = YES
0	RO	X	SATA_RX_STAT_INT_STATUS: SATA rx_stat interrupt status from the SATA pad 1 = rx_stat interrupt is pending 0 = rx_stat interrupt is not pending 0 = NONE 1 = PENDING

21.1.5.3 SATA_AUX_RX_STAT_SET_0

RX_STAT_SET Register

Offset: 0x1110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	SATA_DEVSLP_INT_SET: SATA devslp interrupt is set (this bit is auto-cleared). Set the interrupt status bit to '1' when register is written. The read back value is always '0'.
1	0x0	SATA_DEV_ATTEN_INT_SET: SATA dev_atten interrupt is set (this bit is auto-cleared). Set the interrupt status bit to '1' when register is written. The read back value is always '0'.
0	0x0	SATA_RX_STAT_INT_SET: SATA rx_stat interrupt is set (this bit is auto-cleared). Set the interrupt status bit to '1' when register is written. The read back value is always '0'.

21.1.5.4 SATA_AUX_RX_STAT_CLR_0

RX_STAT_CLR Register

Offset: 0x1114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	SATA_DEVSLP_INT_CLR: SATA devslp interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.
1	0x0	SATA_DEV_ATTEN_INT_CLR: SATA dev_atten interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.
0	0x0	SATA_RX_STAT_INT_CLR: SATA rx_stat interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.

21.1.5.5 SATA_AUX_SPARE_CFG0_0

SPARE_CFG0 Register

Offset: 0x1118 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxx0000000xxxxxxxx)

Bit	R/W	Reset	Description
14	RW	0x0	MDAT_TIMER_AFTER_PG_VALID: Indicates that the MDAT timer value in the above field is to be loaded in the SATA register space.
13:8	RW	0x0	MDAT_TIMER_AFTER_PG: MDAT timer value to be updated by the software (PEP driver) before power-ungating the SATA controller.
5:0	RO	X	MDAT_TIMER_BEFORE_PG: MDAT timer value loaded from the SATA register space before SATA is power-gated.

21.1.5.6 SATA_AUX_SPARE_CFG1_0

SPARE_CFG1 Register

Offset: 0x111c | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:0	-65536	SPARE: Spare register bits

21.1.5.7 SATA_AUX_PAD_PLL_CTRL_0_0

SATA PAD PLL Control Register

Offset: 0x1120 | Read/Write: R/W | Reset: 0x3X3X0000 (0bxx11xxx0xx11xxx100000x0000000000)

Bit	R/W	Reset	Description
29:28	RW	0x3	PLL1_REFCLK_NDIV
27	RO	X	PLL1_LOCKDET
24	RW	0x0	PLL1_MODE
21:20	RW	0x3	PLL0_REFCLK_NDIV: Select feedback divider for PLL.
19	RO	X	PLL0_LOCKDET: Status signal indicating whether PLL is locked within the desired resolution. Forced high when the PLL is in test modes enabled by either PLL_BYPASS_EN or PLL_EMULATION_ON. 0 = Not locked 1 = Locked 0 = NOT_LOCKED 1 = LOCKED
16	RW	0x1	PLL0_MODE
15:12	RW	0x0	REFCLK_SEL: 00 = Internal CML reference clock 01 = Internal CMOS reference clock 1x = External reference clock 0 = INT_CML 1 = INT_CMOS 2 = EXT
11	RW	0x0	REFCLK_TERM100
9	RW	0x0	PLL_CKBUFPD_OVRD
8	RW	0x0	PLL_CKBUFPD_M
7	RW	0x0	PLL_CKBUFPD_BL
6	RW	0x0	PLL_CKBUFPD_BR
5	RW	0x0	PLL_CKBUFPD_TL
4	RW	0x0	PLL_CKBUFPD_TR
3	RW	0x0	SPARE_BIT_2: Reserved bit.
2	RW	0x0	PLL_EMULATION_RSTN: Digital reset for clock divider during emulation mode, hold low (reset) and release to High to set divider phase. No effect during normal operation. 0 = Reset hold 1 = Reset released/active 0 = ASSERT 1 = DEASSERT
1	RW	0x0	SPARE_BIT_1: Reserved bit.
0	RW	0x0	SPARE_BIT_0: Reserved bit. 0 = LOW 1 = HIGH

21.1.5.8 SATA_AUX_PAD_PLL_CTRL_1_0

Offset: 0x1124 | Read/Write: R/W | Reset: 0xXX441020 (0bxxxxxxxx010001000x01000000100000)

Bit	R/W	Reset	Description
31:24	RO	X	PLL_MISC_OUT: Reserved.
23:20	RW	0x4	PLL1_CP_CNTL: Charge-pump current control for PLL1.
19:16	RW	0x4	PLL0_CP_CNTL: Charge-pump current control for PLL0.
15	RW	0x0	PLL_BYPASS_EN: Bypass PLL serial output clocks with input reference clock.
13	RW	0x0	PLL_EMULATION_ON: Enable clock bypass for emulation mode.
12	RW	0x1	TCLKOUT_EN: Enable test clock output pads, TSTCLKP/N.
11:8	RW	0x0	TCLKOUT_SEL: Select internal clock source to bring out through the TSTCLKP/N pads.
7	RW	0x0	XDIGCLK4P5_EN: Enable XDIGCLK4P5 clock output to core.
6	RW	0x0	REFCLKBUF_EN: Enable REFCLKBUF clock output to core.
5	RW	0x1	TXCLKREF_EN: Enable TXCLKREF clock to core.
4	RW	0x0	TXCLKREF_SEL: Select the post divider for TXCLKREF clock.
3	RW	0x0	XDIGCLK_EN: Enable XDIGCLK output clock.
2:0	RW	0x0	XDIGCLK_SEL: Select the output frequency of XDIGCLK.

21.1.5.9 SATA_AUX_PAD_PLL_CTRL_2_0

Offset: 0x1128 | Read/Write: R/W | Reset: 0x0000XX80 (0b0000xxxx00000000x0xxxxxx1xx00000)

Bit	R/W	Reset	Description
31:28	RW	0x0	PLL_TEMP_CNTL
23:18	RW	0x0	PLL_BW_CNTL
17:16	RW	0x0	PLL_BGAP_CNTL
15	RO	X	RCAL_DONE: Status signal to indicate calibration status.
14	RW	0x0	RCAL_RESET: Reset the resistor calibration logic.
12:8	RO	X	RCAL_VAL: Setting of current active resistor calibration code
7	RW	0x1	RCAL_BYPASS: Bypass resistor calibration logic.
4:0	RW	0x0	RCAL_CODE: Sets resistor calibration code when logic is bypassed.

21.1.5.10 SATA_AUX_PAD_PLL_CTRL_3_0

Offset: 0x112c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PLL_MISC_CNTL

21.1.5.11 SATA_AUX_PAD_L0_AUX_CTRL_0_0

Offset: 0x1130 | Read/Write: R/W | Reset: 0x00000X08 (0bxxxxxxxxxxxxxxxxxxxxx0xxx00001000)

Bit	R/W	Reset	Description
11	RW	0x0	AUX_TX_RDET_CLK_EN
10:9	RO	X	AUX_RX_IDLE_STATUS
8	RO	X	AUX_TX_RDET_STATUS
7	RW	0x0	AUX_RX_TERM_MODE
6	RW	0x0	AUX_HOLD_EN
5:4	RW	0x0	AUX_RX_IDLE_MODE
3	RW	0x1	AUX_RX_IDLE_EN
2	RW	0x0	AUX_RX_TERM_EN

Bit	R/W	Reset	Description
1	RW	0x0	AUX_TX_RDET_EN
0	RW	0x0	AUX_TX_TERM_EN

21.1.6 DAC/DAP Registers

21.1.6.1 APB_MISC_DAS_DAP_CTRL_SEL_0

DAP Control Register

This is an array of 5 identical register entries; the register fields below apply to each entry.

Offset: 0xc00..0xc13 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
31	0x0	DAP_MS_SEL: This bit is programmed to put a particular DAP in either master or slave mode when two or more DAPs are in bypass mode. 0 = SLAVE 1 = MASTER
30	0x0	DAP_SDATA1_TX_RX: Programs sdata1 in either tx or rx mode when two or more DAPs are in bypass mode. 0 = TX 1 = RX
29	0x0	DAP_SDATA2_RX_TX: Programs sdata2 in either tx or rx mode when two or more DAPs are in bypass mode. 0 = RX 1 = TX
4:0	0x0	DAP_CTRL_SEL: DAP selection bits to select one of the three DACs or one of the five DAPs. 0 = DAC1 1 = DAC2 2 = DAC3 16 = DAP1 17 = DAP2 18 = DAP3 19 = DAP4 20 = DAP5

21.1.6.2 APB_MISC_DAS_DAC_INPUT_DATA_CLK_SEL_0

DAC Input Data Selections

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xc40..0xc4b | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31:28	0x0	DAC_SDATA2_SEL: These bits control the sdata2 input selection for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5
27:24	0x0	DAC_SDATA1_SEL: These bits control the sdata1 input selection for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5
3:0	0x0	DAC_CLK_SEL: These bits control the bit clock and fsync selection for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5

21.1.7 AP Control Registers

Secure Registers

21.1.7.1 APB_MISC_SECURE_REGS_APB_SLAVE_SECURITY_ENABLE_REG0_0

This block of registers is only accessible from a TrustZone® enabled master operating in secure mode.

APB Slave Security Enable Register 0

These registers are used to enable and disable secure access of corresponding APB slaves. If set, the corresponding APB slave is accessed only via secure transactions.

This register may only be accessed in TrustZone secure mode.

Offset: 0xc00 | Read/Write: R/W | Reset: 0x00000000 (0bxx0xxxx00000x0x0000x0000x0x0000x)

Bit	Reset	Description
29	0x0	STM_SECURITY_EN: STM
24	0x0	CEC_SECURITY_EN: CEC
23	0x0	ATOMICS_SECURITY_EN: Atomics
22	0x0	LA_SECURITY_EN: LA
21	0x0	HDA_SECURITY_EN: HDA
20	0x0	SATA_SECURITY_EN: SATA
18	0x0	Unused, reserved.
16	0x0	KFUSE_SECURITY_EN: kfuse
15	0x0	FUSE_SECURITY_EN: Fuse
14	0x0	SE_SECURITY_EN: Security Engine
13	0x0	PMC_SECURITY_EN: PMC
11	0x0	RTC_SECURITY_EN: RTC
10	0x0	CSITE_SECURITY_EN: Core Site
9	0x0	QSPI_SECURITY_EN: QSPI
8	0x0	PWM_SECURITY_EN: PWFM
6	0x0	DTV_SECURITY_EN: DTV
4	0x0	APE_SECURITY_EN: APE
3	0x0	PINMUX_AUX_SECURITY_EN: Pinmux aux registers
2	0x0	SATA_AUX_SECURITY_EN: SATA aux registers
1	0x0	MISC_REGS_SECURITY_EN: PP, SC1x pads and GP registers

21.1.7.2 APB_MISC_SECURE_REGS_APB_SLAVE_SECURITY_ENABLE_REG1_0

APB Slave Security Enable Register 1

This register may only be accessed in TrustZone secure mode.

Offset: 0xc04 | Read/Write: R/W | Reset: 0x00000000 (0b000000000000xxxx00000000xx00xxxx)

Bit	Reset	Description
31	0x0	I2C6_SECURITY_EN: I2C6
30	0x0	DVC_SECURITY_EN: DVC - I2C5
29	0x0	I2C4_SECURITY_EN: I2C4
28	0x0	I2C3_SECURITY_EN: I2C3
27	0x0	I2C2_SECURITY_EN: I2C2
26	0x0	I2C1_SECURITY_EN: I2C1
25	0x0	SPI6_SECURITY_EN: SPI6
24	0x0	SPI5_SECURITY_EN: SPI5

Bit	Reset	Description
23	0x0	SPI4_SECURITY_EN: SPI4
22	0x0	SPI3_SECURITY_EN: SPI3
21	0x0	SPI2_SECURITY_EN: SPI2
20	0x0	SPI1_SECURITY_EN: SPI1
15	0x0	UART_D_SECURITY_EN: UARTD
14	0x0	UART_C_SECURITY_EN: UARTC
13	0x0	UART_B_SECURITY_EN: UARTB
12	0x0	UART_A_SECURITY_EN: UARTA
11	0x0	EMCB_SECURITY_EN: EMCB
10	0x0	MCB_SECURITY_EN: MCB
9	0x0	EMC1_SECURITY_EN: EMC1
8	0x0	MC1_SECURITY_EN: MC1
5	0x0	EMC0_SECURITY_EN: EMC0
4	0x0	MC0_SECURITY_EN: MC0

21.1.7.3 APB_MISC_SECURE_REGS_APB_SLAVE_SECURITY_ENABLE_REG2_0

APB Slave Security Enable Register 2

This register may only be accessed in TrustZone secure mode.

Offset: 0xc08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000xxx0000)

Bit	Reset	Description
16	0x0	DVFS_SECURITY_EN: DVFS
15	0x0	MIPI_CAL_SECURITY_EN: MIPI cal
14	0x0	XUSB_PADCTL_SECURITY_EN: XUSB_PADCTL
13	0x0	XUSB_DEV_SECURITY_EN: XUSB_dev
12	0x0	XUSB_HOST_SECURITY_EN: XUSB_host
11	0x0	APB2JTAG_SECURITY_EN: APB2JTAG
10	0x0	SOC_THERM_SECURITY_EN: SOC_THERM
9	0x0	DP2_SECURITY_EN: DP2
8	0x0	DDS_SECURITY_EN: DDS
7	0x0	MIPIBIF_SECURITY_EN: (Reserved)
3	0x0	SDMMC4_SECURITY_EN: SDMMC4
2	0x0	SDMMC3_SECURITY_EN: SDMMC3
1	0x0	SDMMC2_SECURITY_EN: SDMMC2
0	0x0	SDMMC1_SECURITY_EN: SDMMC1

21.1.8 Pin Mux Selects

Refer to Chapter 9: Multi-Purpose I/O Pins and Pin Multiplexing (Pinmuxing) in this document for details on the Pinmux_aux registers.

21.2 APB DMA Controller

The APB DMA Controller is placed between the AHB Bus and the APB Bus and is a master on both buses.

The APB DMA Controller is used for block data transfers from a source location to the destination location. The source may be DRAM or IRAM, and the destination location could be devices placed on APB Bus; or vice versa. DMA transfers are done without any processor intervention other than register writes needed to program the parameters for a particular transfer, and accesses needed to handle any interrupts.

The APB DMA Controller has 32 fully programmable channels, which can all transfer data concurrently.

21.2.1 Features

- DMA master for transfers between external/internal memory and peripheral devices on the APB Bus
- Two modes of operation: single transfer (once) or continuous
- Programmable burst sizes of 1, 4, or 8 words
- Maximum transfer size is 1 GB per channel, with the minimum size being one word
- Programmable APB bus widths of 8, 16, and 32 bits
- Separate AHB and APB start addresses
- Per channel trigger and flow control mechanism support
- Channel to channel trigger support, i.e., ability to link up channels to start at the end of another channel's transfer, allowing scattering/gathering of physical memory.
- Interrupt generation at the completion of channel transfer
- Interrupts per channel can be routed to the CPU or BPMP-Lite
- Ability to hold processor until transfer is done
- Wrap mode supported for all channels in Once mode
- Ping-Pong feature supported in continuous mode
- Weighted round robin arbitration among channels at burst granularity. The weights for each channel can be set.
- Runs on APB clock. The APB clock generally runs at 1:2:2 or 1:2:3 clock ratios (sclk : hclk : pclk).
- APB DMA secure access feature allows configuration of secure and non-secure areas
- Address wraparound
- Global or individual channel pausing

21.2.2 Functionality

There are 32 channels in APB DMA. An APB DMA channel can transfer specified portions of data from an AHB address space (can be iRAMs or DRAMs on the MC) to an APB address space. APB DMA follows a simple round robin arbitration scheme, starting with channel 0.

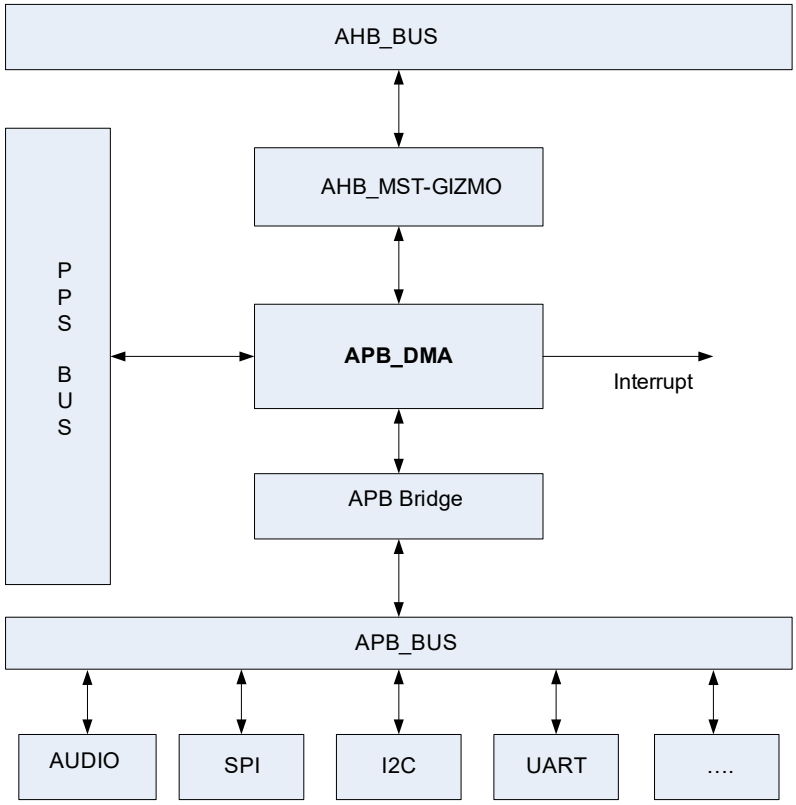
Each channel can have independent burst transfer sizes programmed to one word, four words, or eight words.

Each DMA channel supports:

- Enable bit: one for each channel and one global enable bit.
- Interrupt Enable at the end of transfer with ability to mask or route to desired processor.
- Ability to hold off a processor. The Processor that writes into this bit will be held until transfer is completed.

- Direction bit to determine the direction of transfer--AHB to APB or APB to AHB.
- Trigger and Flow selects. These are addition controls apart from channel enable, on which the transfer depends. Trigger is used to start a channel on some event to start the transfer and flow is used to proceed with every new burst transfer. These events are under either Software or Hardware control.
- Wrap feature wraps the LS nibble of the address back to ‘b0000 instead of incrementing to the next address if the required number of words has been transmitted.
- APB bus width is configurable whereas the AHB bus width is fixed to 32 bits. The APB bus can be 8, 16, or 32 bits wide. For an AHB burst of 8 (burst is always with regard to words) and an APB bus size of 16, there are: 8 words transferred to the APB side, or 16 halfwords transferred.
- Double Buffering Mode makes the APB DMA Burst Address reset to AHB Base Address after every even (2nd, 4th, 6th, etc.) APB DMA Transfer. This mode is used only along with continuous mode (once bit 0).
- Separate AHB start address and APB start address.
- Ability to delay the burst rate with the help of a global counter which can be programmed to desired value, which equals number of clocks delay required between each burst.

Figure 49: APB-DMA Block Diagram



21.2.3 APB DMA Secure Access

APB DMA secure access is determined by the channel security enable bits in the APB DMA Security register. A particular channel is secured by enabling the respective bit field section. Whether through the CPU or APB DMA, accesses are blocked if the SECURITY_EN bit of that region is enabled and the incoming access is non-secured.

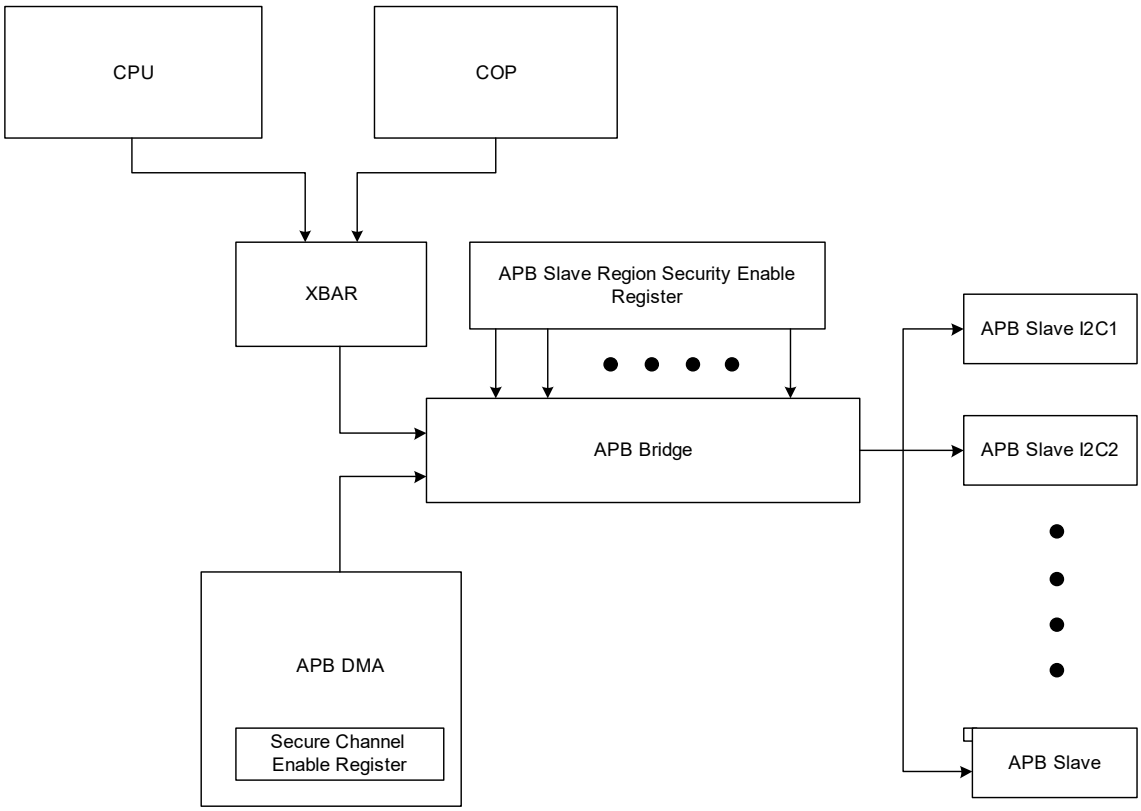
The APBDMA_SECURITY_REG register contains the fields CH_<0-31>_SECURITY_EN, which indicate which channel is secure. Write and read transactions to this register must always be secure.

If a channel is designated as secure, its registers must be written and read out in a secure manner. A non-secure write attempt will not affect any of the secure channel’s registers and a non-secure read attempt will not generate any read data.

If a channel is non-secure, its registers can be written or read either in a secure or a non-secure manner.

The following figure shows the security flow.

Figure 50: Security Flow Diagram



21.2.4 Programming Guidelines

Follow these guidelines when programming the APB DMA Controller:

The DMA transfer size should be in multiples of 4 bytes only.

The APB burst size and the FIFO trigger levels (in the modules) need to be programmed such that they do not lead to an overflow/underflow of the FIFO in the APB client.

All registers of a channel should be programmed before the Channel Enable bit in the channel's control register is set as follows:

1. Program the AHB Starting Address and APB Starting address in the *AHB_PTR and *APB_PTR registers.
2. Program the required AHB BURST size, WRAP word window size, and AHB_DATA_SWAP (byte swapping) option in the *AHB_SWQ register. The AHB BUS WIDTH is fixed to a 32-bit bus.
3. Program the required APB_BUS_WIDTH (as the peripheral), APB_DATA_SWAP (byte swapping) option, and WRAP word window size in the *APB_SEQ register.
4. Program the number of words to be transferred in the *WCOUNT register.
5. Program the trigger in the CHANNEL_*_CSRE register.
6. Program the Interrupt option, Hold Processor option, DMA transfer direction, Transfer Mode, and Flow Enable in the CHANNEL_*_CSR register.
7. Program the Global Enable (GEN) bit in the APB_DMA Command Register. The GEN bit can also be programmed before programming the channel registers.
8. Whenever the channel's ENB bit is enabled, the DMA starts the data transfer.

9. Each channel's status is observed by polling the APB_DMA Status Register. The number of words remaining to be transferred will be in the WORD_TRANSFER register.
10. The Tx/Rx Flow/Trigger requesters are programmed in the APB-DMA Requester Assignments Registers.
11. The busy (BSY) bit is set as soon as a DMA channel is enabled and is cleared after the transfer is completed.

Pausing and Ending of Data Transfers

- Clearing the Global Enable (GEN) bit causes the transfers that are in progress to be paused. Setting the GEN bit resumes the transfers. A channel should never be disabled when the DMA is in global pause.
- Setting the CHANNEL_PAUSE bit in the CSRE register pauses data transfers on that channel. If the CHANNEL_PAUSE bit of a particular channel is enabled during a data transfer, the data transfer is paused only on this channel after the completion of the current burst. It can be resumed by setting this bit to 0.
- If a channel's ENB is disabled independently while a transfer is in progress, the transfer ends after completing any burst sequence that is in progress.

Interrupts

- Interrupts are write-1-to-clear, i.e., an interrupt bit is cleared when the value of write data corresponding to the bit position of the interrupt bit is 1.
- An interrupt is generated when the last data is accepted by the AHB slave while writing on the AHB bus and once the last data is placed on APB bus while reading from AHB bus. Note that completion of an AHB write does not guarantee the data is written to DRAM. Refer to Chapter 19: AHB of this document for how data is flushed from AHB to the MC for this master.

Wraparound

Wraparound does not occur in between bursts; it is only after completing a burst that the address wraps around. Wraparound is possible for both AHB and APB addresses.

The programming of the AHB wraparound needs to account for the address that is programmed in the AHB start address and the burst size.

Usually, APB clients have a buffer of a fixed size. If the APB addresses are incremented after each word transfer, it is likely that the address would cross the address limit of that client and go into the address range of another client. To avoid this, the wrap feature is used. The default wraparound on the APB side is wrapping on 1 word. It prevents unnecessary address switching on the APB.

With the address wrap feature enabled, the LS nibble of the address wraps back to 0 after completion of the programmed number of words instead of incrementing the address.

Example: AHB burst size = 4 words, AHB start address = 0x4000_0000, AHB wrap on 32 words

In this case, the AHB addresses would be:

0x4000_0000, 0x4000_0004, 0x4000_0008, 0x4000_000C

…….

0x4000_0070, 0x4000_0074, 0x4000_0078, 0x4000_007C

When the address wrap feature is disabled, the addresses are incremented after each word transfer. This helps transfer data from/to contiguous locations.

Flow Control

When flow control is enabled, the number of words to be transferred must always be a multiple of the burst size/APB trigger level. Without flow control the following are possible:

- If a burst of 8 is requested with an address that is not aligned to an 8-word boundary, the DMA will do a single burst until it aligns itself to the 8-word boundary and then starts issuing burst requests.

- If a burst of 8 is requested and at any point of time the transfer size goes below 8, the DMA completes the remaining transfers with a burst of 4 or 1, whichever is possible.
- If a burst of 4 is requested with an address that is not aligned to a 4-word boundary, the DMA transfers a single burst until it aligns itself to the 4-word boundary.
- If a burst of 4 is requested and at any point of time the transfer size goes below 4, the DMA completes the remaining transfers with a burst of 1.

Continuous Mode

In continuous mode, when a transfer is in progress, the APB and AHB addresses and the WCOUNT fields can be reprogrammed in the middle of a data transfer. When the current transfer completes and a new transfer starts, it picks up the newly programmed values. All new programming must complete before the channel completes the data transfer. To be sure about this, CHANNEL_PAUSE can be used.

2x Mode

In 2x mode, the reprogramming can be done only on the second half of the data transfer.

21.2.5 APB DMA Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

21.2.5.1 APBDMA_COMMAND_0

APB-DMA Command Register

The Global Enable (GEN) bit in the command register enables the APB DMA. Clearing this bit causes active DMA transfers to be paused. Pending bus transactions (ongoing burst) will be completed, and no new transactions will be initiated. Setting this bit again resumes the transfers. Disabling this bit will disable the channel register access.

Note that the power on reset value for the APB DMA Global Enable is 0. This bit must be written to 1 before any APB DMA transactions can begin.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	GEN: Enables Global APB-DMA 0 = DISABLE 1 = ENABLE

21.2.5.2 APBDMA_STATUS_0

APB-DMA Status Register

The Busy bits in the Status Register indicate which (if any) of the APB DMA channels have active pending APB DMA transfers. Note that Busy bits remain active in APB DMA transfers that are started in continuous (repetitive) mode until the enable bit for the APB DMA channel is cleared to 0 (by the software). Read-only flags are set/cleared by hardware.

Offset: 0x4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	BSY_31: DMA channel 31 status 0 = NOT_BUSY 1 = BUSY
30	X	BSY_30: DMA channel 30 status 0 = NOT_BUSY 1 = BUSY
29	X	BSY_29: DMA channel 29 status 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
28	X	BSY_28: DMA channel 28 status 0 = NOT_BUSY 1 = BUSY
27	X	BSY_27: DMA channel 27 status 0 = NOT_BUSY 1 = BUSY
26	X	BSY_26: DMA channel 26 status 0 = NOT_BUSY 1 = BUSY
25	X	BSY_25: DMA channel 25 status 0 = NOT_BUSY 1 = BUSY
24	X	BSY_24: DMA channel 24 status 0 = NOT_BUSY 1 = BUSY
23	X	BSY_23: DMA channel 23 status 0 = NOT_BUSY 1 = BUSY
22	X	BSY_22: DMA channel 22 status 0 = NOT_BUSY 1 = BUSY
21	X	BSY_21: DMA channel 21 status 0 = NOT_BUSY 1 = BUSY
20	X	BSY_20: DMA channel 20 status 0 = NOT_BUSY 1 = BUSY
19	X	BSY_19: DMA channel 19 status 0 = NOT_BUSY 1 = BUSY
18	X	BSY_18: DMA channel 18 status 0 = NOT_BUSY 1 = BUSY
17	X	BSY_17: DMA channel 17 status 0 = NOT_BUSY 1 = BUSY
16	X	BSY_16: DMA channel 16 status 0 = NOT_BUSY 1 = BUSY
15	X	BSY_15: DMA channel 15 status 0 = NOT_BUSY 1 = BUSY
14	X	BSY_14: DMA channel 14 status 0 = NOT_BUSY 1 = BUSY
13	X	BSY_13: DMA channel 13 status 0 = NOT_BUSY 1 = BUSY
12	X	BSY_12: DMA channel 12 status 0 = NOT_BUSY 1 = BUSY
11	X	BSY_11: DMA channel 11 status 0 = NOT_BUSY 1 = BUSY
10	X	BSY_10: DMA channel 10 status 0 = NOT_BUSY 1 = BUSY
9	X	BSY_9: DMA channel 9 status 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
8	X	BSY_8: DMA channel 8 status 0 = NOT_BUSY 1 = BUSY
7	X	BSY_7: DMA channel 7 status 0 = NOT_BUSY 1 = BUSY
6	X	BSY_6: DMA channel 6 status 0 = NOT_BUSY 1 = BUSY
5	X	BSY_5: DMA channel 5 status 0 = NOT_BUSY 1 = BUSY
4	X	BSY_4: DMA channel 4 status 0 = NOT_BUSY 1 = BUSY
3	X	BSY_3: DMA channel 3 status 0 = NOT_BUSY 1 = BUSY
2	X	BSY_2: DMA channel 2 status 0 = NOT_BUSY 1 = BUSY
1	X	BSY_1: DMA channel 1 status 0 = NOT_BUSY 1 = BUSY
0	X	BSY_0: DMA channel 0 status 0 = NOT_BUSY 1 = BUSY

21.2.5.3 APBDMA_CNTRL_REG_0

APB-DMA Counter Register

The APB DMA Counter is used to slow down the request rates on some APB DMA channels. The APB DMA Channel Counter Enable register stores bits that configure which (if any) channel(s) should be throttled (bits [31:0] of channel counter enable register correspond to the 32 APB DMA channels).

The APB DMA Counter initial/reload count value is programmed in the APB DMA Counter Register (bits[15:0]).The APB DMA Counter is loaded with this initial/reload count value whenever the APB DMA Counter Register is written, and is re-loaded to this saved initial count value on a burst complete (programmable). The APB DMA Counter value will decrement whenever an APB DMA burst complete, and the current count value is non-zero, and any of the bits [31:16] are set.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	COUNT_VALUE: DMA COUNT Value.

21.2.5.4 APBDMA_IRQ_STA_CPU_0

APB-DMA CPU IRQ STATUS Register

Gathers all the after-masking CPU directed IRQ status bits

Offset: 0x14 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Gathers all the after-masking CPU directed IRQ status bits from channel 31 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	X	CH30: Gathers all the after-masking CPU directed IRQ status bits from channel 30 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
29	X	CH29: Gathers all the after-masking CPU directed IRQ status bits from channel 29 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
28	X	CH28: Gathers all the after-masking CPU directed IRQ status bits from channel 28 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
27	X	CH27: Gathers all the after-masking CPU directed IRQ status bits from channel 27 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
26	X	CH26: Gathers all the after-masking CPU directed IRQ status bits from channel 26 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
25	X	CH25: Gathers all the after-masking CPU directed IRQ status bits from channel 25 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
24	X	CH24: Gathers all the after-masking CPU directed IRQ status bits from channel 24 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
23	X	CH23: Gathers all the after-masking CPU directed IRQ status bits from channel 23 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
22	X	CH22: Gathers all the after-masking CPU directed IRQ status bits from channel 22 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
21	X	CH21: Gathers all the after-masking CPU directed IRQ status bits from channel 21 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
20	X	CH20: Gathers all the after-masking CPU directed IRQ status bits from channel 20 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
19	X	CH19: Gathers all the after-masking CPU directed IRQ status bits from channel 19 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	X	CH18: Gathers all the after-masking CPU directed IRQ status bits from channel 18 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
17	X	CH17: Gathers all the after-masking CPU directed IRQ status bits from channel 17 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
16	X	CH16: Gathers all the after-masking CPU directed IRQ status bits from channel 16 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
15	X	CH15: Gathers all the after-masking CPU directed IRQ status bits from channel 15 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking CPU directed IRQ status bits from channel 14 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking CPU directed IRQ status bits from channel 13 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking CPU directed IRQ status bits from channel 12 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking CPU directed IRQ status bits from channel 11 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking CPU directed IRQ status bits from channel 10 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking CPU directed IRQ status bits from channel 9 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking CPU directed IRQ status bits from channel 8 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking CPU directed IRQ status bits from channel 7 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	X	CH6: Gathers all the after-masking CPU directed IRQ status bits from channel 6 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
5	X	CH5: Gathers all the after-masking CPU directed IRQ status bits from channel 5 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
4	X	CH4: Gathers all the after-masking CPU directed IRQ status bits from channel 4 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking CPU directed IRQ status bits from channel 3 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
2	X	CH2: Gathers all the after-masking CPU directed IRQ status bits from channel 2 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking CPU directed IRQ status bits from channel 1 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking CPU directed IRQ status bits from channel 0 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

21.2.5.5 APBDMA_IRQ_STA_COP_0

APB-DMA COP IRQ STATUS Register

Gathers all the after-masking COP directed IRQ status bits

Offset: 0x18 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Gathers all the after-masking COP directed IRQ status bits from channel 31 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
30	X	CH30: Gathers all the after-masking COP directed IRQ status bits from channel 30 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
29	X	CH29: Gathers all the after-masking COP directed IRQ status bits from channel 29 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	X	CH28: Gathers all the after-masking COP directed IRQ status bits from channel 28 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
27	X	CH27: Gathers all the after-masking COP directed IRQ status bits from channel 27 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
26	X	CH26: Gathers all the after-masking COP directed IRQ status bits from channel 26 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
25	X	CH25: Gathers all the after-masking COP directed IRQ status bits from channel 25 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
24	X	CH24: Gathers all the after-masking COP directed IRQ status bits from channel 24 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
23	X	CH23: Gathers all the after-masking COP directed IRQ status bits from channel 23 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
22	X	CH22: Gathers all the after-masking COP directed IRQ status bits from channel 22 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
21	X	CH21: Gathers all the after-masking COP directed IRQ status bits from channel 21 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
20	X	CH20: Gathers all the after-masking COP directed IRQ status bits from channel 20 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
19	X	CH19: Gathers all the after-masking COP directed IRQ status bits from channel 19 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
18	X	CH18: Gathers all the after-masking COP directed IRQ status bits from channel 18 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
17	X	CH17: Gathers all the after-masking COP directed IRQ status bits from channel 17 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
16	X	CH16: Gathers all the after-masking COP directed IRQ status bits from channel 16 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
15	X	CH15: Gathers all the after-masking COP directed IRQ status bits from channel 15 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking COP directed IRQ status bits from channel 14 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking COP directed IRQ status bits from channel 13 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking COP directed IRQ status bits from channel 12 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking COP directed IRQ status bits from channel 11 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking COP directed IRQ status bits from channel 10 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking COP directed IRQ status bits from channel 9 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking COP directed IRQ status bits from channel 8 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking COP directed IRQ status bits from channel 7 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
6	X	CH6: Gathers all the after-masking COP directed IRQ status bits from channel 6 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
5	X	CH5: Gathers all the after-masking COP directed IRQ status bits from channel 5 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	X	CH4: Gathers all the after-masking COP directed IRQ status bits from channel 4 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking COP directed IRQ status bits from channel 3 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
2	X	CH2: Gathers all the after-masking COP directed IRQ status bits from channel 2 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking COP directed IRQ status bits from channel 1 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking COP directed IRQ status bits from channel 0 0 = No IRQ pending 1 = IRQ pending 0 = DISABLE 1 = ENABLE

21.2.5.6 APBDMA_IRQ_MASK_0

APB-DMA IRQ MASK Register

Allows the IRQ to propagate when enabled, this is the ANDed result of set and clear.

Offset: 0x1c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Each bit allows the associated channel31 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
30	X	CH30: Each bit allows the associated channel30 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
29	X	CH29: Each bit allows the associated channel29 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
28	X	CH28: Each bit allows the associated channel28 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
27	X	CH27: Each bit allows the associated channel27 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
26	X	CH26: Each bit allows the associated channel26 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
25	X	CH25: Each bit allows the associated channel25 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
24	X	CH24: Each bit allows the associated channel24 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
23	X	CH23: Each bit allows the associated channel23 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	X	CH22: Each bit allows the associated channel22 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
21	X	CH21: Each bit allows the associated channel21 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
20	X	CH20: Each bit allows the associated channel20 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
19	X	CH19: Each bit allows the associated channel19 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
18	X	CH18: Each bit allows the associated channel18 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
17	X	CH17: Each bit allows the associated channel17 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
16	X	CH16: Each bit allows the associated channel16 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
15	X	CH15: Each bit allows the associated channel15 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
14	X	CH14: Each bit allows the associated channel14 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
13	X	CH13: Each bit allows the associated channel13 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
12	X	CH12: Each bit allows the associated channel12 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
11	X	CH11: Each bit allows the associated channel11 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
10	X	CH10: Each bit allows the associated channel10 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
9	X	CH9: Each bit allows the associated channel9 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
8	X	CH8: Each bit allows the associated channel8 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
7	X	CH7: Each bit allows the associated channel7 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
6	X	CH6: Each bit allows the associated channel6 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
5	X	CH5: Each bit allows the associated channel5 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
4	X	CH4: Each bit allows the associated channel4 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
3	X	CH3: Each bit allows the associated channel3 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	CH2: Each bit allows the associated channel2 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
1	X	CH1: Each bit allows the associated channel1 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
0	X	CH0: Each bit allows the associated channel0 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

21.2.5.7 APBDMA_IRQ_MASK_SET_0

APB-DMA IRQ MASK SET Register

Offset: 0x20 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
30	0x0	CH30: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
29	0x0	CH29: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
28	0x0	CH28: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
27	0x0	CH27: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
26	0x0	CH26: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
25	0x0	CH25: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
24	0x0	CH24: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
23	0x0	CH23: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x0	CH22: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
21	0x0	CH21: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
20	0x0	CH20: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
19	0x0	CH19: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
18	0x0	CH18: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
17	0x0	CH17: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
16	0x0	CH16: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
15	0x0	CH15: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
14	0x0	CH14: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
13	0x0	CH13: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
12	0x0	CH12: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
11	0x0	CH11: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x0	CH10: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
9	0x0	CH9: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
8	0x0	CH8: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
7	0x0	CH7: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
6	0x0	CH6: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
5	0x0	CH5: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
3	0x0	CH3: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Sets the Mask Register 0 = Disable the IRQ 1 = Enable the IRQ 0 = DISABLE 1 = ENABLE

21.2.5.8 APBDMA_IRQ_MASK_CLR_0

APB-DMA IRQ MASK CLEAR Register

Offset: 0x24 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31: Clears the Mask Register 0 = DISABLE 1 = ENABLE
30	0x0	CH30: Clears the Mask Register 0 = DISABLE 1 = ENABLE
29	0x0	CH29: Clears the Mask Register 0 = DISABLE 1 = ENABLE
28	0x0	CH28: Clears the Mask Register 0 = DISABLE 1 = ENABLE
27	0x0	CH27: Clears the Mask Register 0 = DISABLE 1 = ENABLE
26	0x0	CH26: Clears the Mask Register 0 = DISABLE 1 = ENABLE
25	0x0	CH25: Clears the Mask Register 0 = DISABLE 1 = ENABLE
24	0x0	CH24: Clears the Mask Register 0 = DISABLE 1 = ENABLE
23	0x0	CH23: Clears the Mask Register 0 = DISABLE 1 = ENABLE
22	0x0	CH22: Clears the Mask Register 0 = DISABLE 1 = ENABLE
21	0x0	CH21: Clears the Mask Register 0 = DISABLE 1 = ENABLE
20	0x0	CH20: Clears the Mask Register 0 = DISABLE 1 = ENABLE
19	0x0	CH19: Clears the Mask Register 0 = DISABLE 1 = ENABLE
18	0x0	CH18: Clears the Mask Register 0 = DISABLE 1 = ENABLE
17	0x0	CH17: Clears the Mask Register 0 = DISABLE 1 = ENABLE
16	0x0	CH16: Clears the Mask Register 0 = DISABLE 1 = ENABLE
15	0x0	CH15: Clears the Mask Register 0 = DISABLE 1 = ENABLE
14	0x0	CH14: Clears the Mask Register 0 = DISABLE 1 = ENABLE
13	0x0	CH13: Clears the Mask Register 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	CH12: Clears the Mask Register 0 = DISABLE 1 = ENABLE
11	0x0	CH11: Clears the Mask Register 0 = DISABLE 1 = ENABLE
10	0x0	CH10: Clears the Mask Register 0 = DISABLE 1 = ENABLE
9	0x0	CH9: Clears the Mask Register 0 = DISABLE 1 = ENABLE
8	0x0	CH8: Clears the Mask Register 0 = DISABLE 1 = ENABLE
7	0x0	CH7: Clears the Mask Register 0 = DISABLE 1 = ENABLE
6	0x0	CH6: Clears the Mask Register 0 = DISABLE 1 = ENABLE
5	0x0	CH5: Clears the Mask Register 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Clears the Mask Register 0 = DISABLE 1 = ENABLE
3	0x0	CH3: Clears the Mask Register 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Clears the Mask Register 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Clears the Mask Register 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Clears the Mask Register 0 = DISABLE 1 = ENABLE

21.2.5.9 APBDMA_TRIG_REG_0

APB-DMA Trigger Register

Offset: 0x28 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	TMR2: Trigger select from Timer (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
7	X	TMR1: Trigger select from Timer (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
6	X	XRQ_B: XRQ.B (GPIOB) (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
5	X	XRQ_A: XRQ.A (GPIOA) (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
4	X	SMP_27: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
3	X	SMP_26: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
2	X	SMP_25: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
1	X	SMP_24: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE

21.2.5.10 APBDMA_CHANNEL_TRIG_REG_0

APB-DMA Channel Trigger Registers

Offset: 0x2c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	APB_31: EOC-31 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
30	X	APB_30: EOC-30 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
29	X	APB_29: EOC-29 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
28	X	APB_28: EOC-28 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
27	X	APB_27: EOC-27 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
26	X	APB_26: EOC-26 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
25	X	APB_25: EOC-25 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
24	X	APB_24: EOC-24 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
23	X	APB_23: EOC-23 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
22	X	APB_22: EOC-22 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
21	X	APB_21: EOC-21 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
20	X	APB_20: EOC-20 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
19	X	APB_19: EOC-19 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
18	X	APB_18: EOC-18 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
17	X	APB_17: EOC-17 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
16	X	APB_16: EOC-16 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
15	X	APB_15: EOC-15 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
14	X	APB_14: EOC-14 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
13	X	APB_13: EOC-13 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
12	X	APB_12: EOC-12 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
11	X	APB_11: EOC-11 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
10	X	APB_10: EOC-10 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
9	X	APB_9: EOC-9 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
8	X	APB_8: EOC-8 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
7	X	APB_7: EOC-7 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
6	X	APB_6: EOC-6 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
5	X	APB_5: EOC-5 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
4	X	APB_4: EOC-4 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
3	X	APB_3: EOC-3 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
2	X	APB_2: EOC-2 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
1	X	APB_1: EOC-1 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
0	X	APB_0: EOC-0 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE

21.2.5.11 APBDMA_DMA_STATUS_0

APB-DMA Interrupt Status

Offset: 0x30 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	ISE_EOC_31: DMA Channel31 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
30	X	ISE_EOC_30: DMA Channel30 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
29	X	ISE_EOC_29: DMA Channel29 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
28	X	ISE_EOC_28: DMA Channel28 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
27	X	ISE_EOC_27: DMA Channel27 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
26	X	ISE_EOC_26: DMA Channel26 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
25	X	ISE_EOC_25: DMA Channel25 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
24	X	ISE_EOC_24: DMA Channel24 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
23	X	ISE_EOC_23: DMA Channel23 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
22	X	ISE_EOC_22: DMA Channel22 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
21	X	ISE_EOC_21: DMA Channel21 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
20	X	ISE_EOC_20: DMA Channel20 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
19	X	ISE_EOC_19: DMA Channel19 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
18	X	ISE_EOC_18: DMA Channel18 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
17	X	ISE_EOC_17: DMA Channel17 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
16	X	ISE_EOC_16: DMA Channel16 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
15	X	ISE_EOC_15: DMA Channel15 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
14	X	ISE_EOC_14: DMA Channel14 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
13	X	ISE_EOC_13: DMA Channel13 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
12	X	ISE_EOC_12: DMA Channel12 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
11	X	ISE_EOC_11: DMA Channel11 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
10	X	ISE_EOC_10: DMA Channel10 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
9	X	ISE_EOC_9: DMA Channel9 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
8	X	ISE_EOC_8: DMA Channel8 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
7	X	ISE_EOC_7: DMA Channel7 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
6	X	ISE_EOC_6: DMA Channel6 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
5	X	ISE_EOC_5: DMA Channel5 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
4	X	ISE_EOC_4: DMA Channel4 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
3	X	ISE_EOC_3: DMA Channel3 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
2	X	ISE_EOC_2: DMA Channel2 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
1	X	ISE_EOC_1: DMA Channel1 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
0	X	ISE_EOC_0: DMA Channel0 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE

21.2.5.12 APBDMA_CHANNEL_EN_REG_0

APB-DMA Channel Counter Enable Registers

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31_CNT_EN: Enable the Channel31 count 0 = DISABLE 1 = ENABLE
30	0x0	CH30_CNT_EN: Enable the Channel30 count 0 = DISABLE 1 = ENABLE
29	0x0	CH29_CNT_EN: Enable the Channel29 count 0 = DISABLE 1 = ENABLE
28	0x0	CH28_CNT_EN: Enable the Channel28 count 0 = DISABLE 1 = ENABLE
27	0x0	CH27_CNT_EN: Enable the Channel27 count 0 = DISABLE 1 = ENABLE
26	0x0	CH26_CNT_EN: Enable the Channel26 count 0 = DISABLE 1 = ENABLE
25	0x0	CH25_CNT_EN: Enable the Channel25 count 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	0x0	CH24_CNT_EN: Enable the Channel24 count 0 = DISABLE 1 = ENABLE
23	0x0	CH23_CNT_EN: Enable the Channel23 count 0 = DISABLE 1 = ENABLE
22	0x0	CH22_CNT_EN: Enable the Channel22 count 0 = DISABLE 1 = ENABLE
21	0x0	CH21_CNT_EN: Enable the Channel21 count 0 = DISABLE 1 = ENABLE
20	0x0	CH20_CNT_EN: Enable the Channel20 count 0 = DISABLE 1 = ENABLE
19	0x0	CH19_CNT_EN: Enable the Channel19 count 0 = DISABLE 1 = ENABLE
18	0x0	CH18_CNT_EN: Enable the Channel18 count 0 = DISABLE 1 = ENABLE
17	0x0	CH17_CNT_EN: Enable the Channel17 count 0 = DISABLE 1 = ENABLE
16	0x0	CH16_CNT_EN: Enable the Channel16 count 0 = DISABLE 1 = ENABLE
15	0x0	CH15_CNT_EN: Enable the Channel15 count 0 = DISABLE 1 = ENABLE
14	0x0	CH14_CNT_EN: Enable the Channel14 count 0 = DISABLE 1 = ENABLE
13	0x0	CH13_CNT_EN: Enable the Channel13 count 0 = DISABLE 1 = ENABLE
12	0x0	CH12_CNT_EN: Enable the Channel12 count 0 = DISABLE 1 = ENABLE
11	0x0	CH11_CNT_EN: Enable the Channel11 count 0 = DISABLE 1 = ENABLE
10	0x0	CH10_CNT_EN: Enable the Channel10 count 0 = DISABLE 1 = ENABLE
9	0x0	CH9_CNT_EN: Enable the Channel9 count 0 = DISABLE 1 = ENABLE
8	0x0	CH8_CNT_EN: Enable the Channel8 count 0 = DISABLE 1 = ENABLE
7	0x0	CH7_CNT_EN: Enable the Channel7 count 0 = DISABLE 1 = ENABLE
6	0x0	CH6_CNT_EN: Enable the Channel6 count 0 = DISABLE 1 = ENABLE
5	0x0	CH5_CNT_EN: Enable the Channel5 count 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	CH4_CNT_EN: Enable the Channel4 count 0 = DISABLE 1 = ENABLE
3	0x0	CH3_CNT_EN: Enable the Channel3 count 0 = DISABLE 1 = ENABLE
2	0x0	CH2_CNT_EN: Enable the Channel2 count 0 = DISABLE 1 = ENABLE
1	0x0	CH1_CNT_EN: Enable the Channel1 count 0 = DISABLE 1 = ENABLE
0	0x0	CH0_CNT_EN: Enable the Channel0 count 0 = DISABLE 1 = ENABLE

21.2.5.13 APBDMA_SECURITY_REG_0

Security enables for each channel.

Secure: TrustZone Protected

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH_31_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
30	0x0	CH_30_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
29	0x0	CH_29_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
28	0x0	CH_28_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
27	0x0	CH_27_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
26	0x0	CH_26_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
25	0x0	CH_25_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
24	0x0	CH_24_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
23	0x0	CH_23_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
22	0x0	CH_22_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
21	0x0	CH_21_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
20	0x0	CH_20_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	CH_19_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
18	0x0	CH_18_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
17	0x0	CH_17_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
16	0x0	CH_16_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
15	0x0	CH_15_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
14	0x0	CH_14_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
13	0x0	CH_13_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
12	0x0	CH_12_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
11	0x0	CH_11_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
10	0x0	CH_10_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
9	0x0	CH_9_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
8	0x0	CH_8_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
7	0x0	CH_7_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
6	0x0	CH_6_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
5	0x0	CH_5_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
4	0x0	CH_4_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
3	0x0	CH_3_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
2	0x0	CH_2_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
1	0x0	CH_1_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE
0	0x0	CH_0_SECURITY_EN: Enables secure channel 0 = DISABLE 1 = ENABLE

21.2.5.14 APBDMA_CHANNEL_SWID_0

SWID[0] for each APBDMA channel. Supports secure writes.

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH_31_SWID:SWID for Channel 31
30	0x0	CH_30_SWID:SWID for Channel 30
29	0x0	CH_29_SWID:SWID for Channel 29
28	0x0	CH_28_SWID:SWID for Channel 28
27	0x0	CH_27_SWID:SWID for Channel 27
26	0x0	CH_26_SWID:SWID for Channel 26
25	0x0	CH_25_SWID:SWID for Channel 25
24	0x0	CH_24_SWID:SWID for Channel 24
23	0x0	CH_23_SWID:SWID for Channel 23
22	0x0	CH_22_SWID:SWID for Channel 22
21	0x0	CH_21_SWID:SWID for Channel 21
20	0x0	CH_20_SWID:SWID for Channel 20
19	0x0	CH_19_SWID:SWID for Channel 19
18	0x0	CH_18_SWID:SWID for Channel 18
17	0x0	CH_17_SWID:SWID for Channel 17
16	0x0	CH_16_SWID:SWID for Channel 16
15	0x0	CH_15_SWID:SWID for Channel 15
14	0x0	CH_14_SWID:SWID for Channel 14
13	0x0	CH_13_SWID:SWID for Channel 13
12	0x0	CH_12_SWID:SWID for Channel 12
11	0x0	CH_11_SWID:SWID for Channel 11
10	0x0	CH_10_SWID:SWID for Channel 10
9	0x0	CH_9_SWID:SWID for Channel 9
8	0x0	CH_8_SWID:SWID for Channel 8
7	0x0	CH_7_SWID:SWID for Channel 7
6	0x0	CH_6_SWID:SWID for Channel 6
5	0x0	CH_5_SWID:SWID for Channel 5
4	0x0	CH_4_SWID:SWID for Channel 4
3	0x0	CH_3_SWID:SWID for Channel 3
2	0x0	CH_2_SWID:SWID for Channel 2
1	0x0	CH_1_SWID:SWID for Channel 1
0	0x0	CH_0_SWID:SWID for Channel 0

21.2.5.15 APBDMA_CHAN_WT_REG0_0

Channel weights for weighted-round-robin arbitration

Offset: 0x44 | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	0x1	WT_CH31: Weight of channel
27:24	0x1	WT_CH30: Weight of channel
23:20	0x1	WT_CH29: Weight of channel
19:16	0x1	WT_CH28: Weight of channel
15:12	0x1	WT_CH27: Weight of channel

Bit	Reset	Description
11:8	0x1	WT_CH26: Weight of channel
7:4	0x1	WT_CH25: Weight of channel
3:0	0x1	WT_CH24: Weight of channel

21.2.5.16 APBDMA_CHAN_WT_REG1_0

Channel weights for weighted-round-robin arbitration

Offset: 0x48 | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	0x1	WT_CH23: Weight of channel
27:24	0x1	WT_CH22: Weight of channel
23:20	0x1	WT_CH21: Weight of channel
19:16	0x1	WT_CH20: Weight of channel
15:12	0x1	WT_CH19: Weight of channel
11:8	0x1	WT_CH18: Weight of channel
7:4	0x1	WT_CH17: Weight of channel
3:0	0x1	WT_CH16: Weight of channel

21.2.5.17 APBDMA_CHAN_WT_REG2_0

Channel weights for weighted-round-robin arbitration

Offset: 0x4c | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	0x1	WT_CH15: Weight of channel
27:24	0x1	WT_CH14: Weight of channel
23:20	0x1	WT_CH13: Weight of channel
19:16	0x1	WT_CH12: Weight of channel
15:12	0x1	WT_CH11: Weight of channel
11:8	0x1	WT_CH10: Weight of channel
7:4	0x1	WT_CH9: Weight of channel
3:0	0x1	WT_CH8: Weight of channel

21.2.5.18 APBDMA_CHAN_WT_REG3_0

Channel weights for weighted-round-robin arbitration

Offset: 0x50 | Read/Write: R/W | Reset: 0x11111111 (0b00010001000100010001000100010001)

Bit	Reset	Description
31:28	0x1	WT_CH7: Weight of channel
27:24	0x1	WT_CH6: Weight of channel
23:20	0x1	WT_CH5: Weight of channel
19:16	0x1	WT_CH4: Weight of channel
15:12	0x1	WT_CH3: Weight of channel
11:8	0x1	WT_CH2: Weight of channel
7:4	0x1	WT_CH1: Weight of channel
3:0	0x1	WT_CH0: Weight of channel

21.2.5.19 APBDMA_CHANNEL_SWID1_0

SWID[1] indicating secure ASID. Supports secure writes.

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH_31_SWID_1: SWID for Channel 31
30	0x0	CH_30_SWID_1: SWID for Channel 30
29	0x0	CH_29_SWID_1: SWID for Channel 29
28	0x0	CH_28_SWID_1: SWID for Channel 28
27	0x0	CH_27_SWID_1: SWID for Channel 27
26	0x0	CH_26_SWID_1: SWID for Channel 26
25	0x0	CH_25_SWID_1: SWID for Channel 25
24	0x0	CH_24_SWID_1: SWID for Channel 24
23	0x0	CH_23_SWID_1: SWID for Channel 23
22	0x0	CH_22_SWID_1: SWID for Channel 22
21	0x0	CH_21_SWID_1: SWID for Channel 21
20	0x0	CH_20_SWID_1: SWID for Channel 20
19	0x0	CH_19_SWID_1: SWID for Channel 19
18	0x0	CH_18_SWID_1: SWID for Channel 18
17	0x0	CH_17_SWID_1: SWID for Channel 17
16	0x0	CH_16_SWID_1: SWID for Channel 16
15	0x0	CH_15_SWID_1: SWID for Channel 15
14	0x0	CH_14_SWID_1: SWID for Channel 14
13	0x0	CH_13_SWID_1: SWID for Channel 13
12	0x0	CH_12_SWID_1: SWID for Channel 12
11	0x0	CH_11_SWID_1: SWID for Channel 11
10	0x0	CH_10_SWID_1: SWID for Channel 10
9	0x0	CH_9_SWID_1: SWID for Channel 9
8	0x0	CH_8_SWID_1: SWID for Channel 8
7	0x0	CH_7_SWID_1: SWID for Channel 7
6	0x0	CH_6_SWID_1: SWID for Channel 6
5	0x0	CH_5_SWID_1: SWID for Channel 5
4	0x0	CH_4_SWID_1: SWID for Channel 4
3	0x0	CH_3_SWID_1: SWID for Channel 3
2	0x0	CH_2_SWID_1: SWID for Channel 2
1	0x0	CH_1_SWID_1: SWID for Channel 1
0	0x0	CH_0_SWID_1: SWID for Channel 0

21.2.6 APB DMA Channel Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Each of the 32 APB DMA channels has its own set of 10 APB DMA registers. Each channel has 64 bytes of address space. This subsection defines one complete set of APB DMA channel registers. The register spaces per APB DMA channel are listed in the table below.

Table 104: APB DMA Channel Register Mapping

Register Space	Channel Registers
0x0 – 0x3f	Channel 0 APB DMA Channel Registers
0x40 – 0x7f	Channel 1 APB DMA Channel Registers

Table 104: APB DMA Channel Register Mapping

Register Space	Channel Registers
0x80 – 0xbf	Channel 2 APB DMA Channel Registers
0xc0 – 0xff	Channel 3 APB DMA Channel Registers
0x100 – 0x13f	Channel 4 APB DMA Channel Registers
0x140 – 0x17f	Channel 5 APB DMA Channel Registers
0x180 – 0x1bf	Channel 6 APB DMA Channel Registers
0x1c0 – 0x1ff	Channel 7 APB DMA Channel Registers
0x200 – 0x23f	Channel 8 APB DMA Channel Registers
0x240 – 0x27f	Channel 9 APB DMA Channel Registers
0x280 – 0x2bf	Channel 10 APB DMA Channel Registers
0x2c0 – 0x2ff	Channel 11 APB DMA Channel Registers
0x300 – 0x33f	Channel 12 APB DMA Channel Registers
0x340 – 0x37f	Channel 13 APB DMA Channel Registers
0x380 – 0x3bf	Channel 14 APB DMA Channel Registers
0x3c0 – 0x3ff	Channel 15 APB DMA Channel Registers
0x400 – 0x43f	Channel 16 APB DMA Channel Registers
0x440 – 0x47f	Channel 17 APB DMA Channel Registers
0x480 – 0x4bf	Channel 18 APB DMA Channel Registers
0x4c0 – 0x4ff	Channel 19 APB DMA Channel Registers
0x500 – 0x53f	Channel 20 APB DMA Channel Registers
0x540 – 0x57f	Channel 21 APB DMA Channel Registers
0x580 – 0x5bf	Channel 22 APB DMA Channel Registers
0x5c0 – 0x5ff	Channel 23 APB DMA Channel Registers
0x600 – 0x63f	Channel 24 APB DMA Channel Registers
0x640 – 0x67f	Channel 25 APB DMA Channel Registers
0x680 – 0x6bf	Channel 26 APB DMA Channel Registers
0x6c0 – 0x6ff	Channel 27 APB DMA Channel Registers
0x700 – 0x73f	Channel 28 APB DMA Channel Registers
0x740 – 0x77f	Channel 29 APB DMA Channel Registers
0x780 – 0x7bf	Channel 30 APB DMA Channel Registers
0x7c0 – 0x7ff	Channel 31 APB DMA Channel Registers

21.2.6.1 APBDMACHAN_CHANNEL_n_CSR_0

APB-DMA-n Control Register

There are 32 APB DMA Channel Control Registers, one per channel (n = 0 through 31).

Writing a 1 to bit [31] of an APB DMA Channel Control Register will initiate the APB DMA transfer. Because this action will depend on some values programmed in the other registers, it is recommended that the registers in the address space in the required APB DMA channel be programmed before writing into control register.

In "Once" mode, the channel is disabled after the APB DMA transfer has completed. When the channel's transfer completes, an interrupt will be sent if the IE.EOC bit is set. If the transfer requires a trigger or flow, then the corresponding trigger selected by the "TRIG_SEL" or "REQ_SEL" field must become active respectively before the channel can starts its transfer. If both Trigger and Flow bits are set, both conditions must be met before the channel starts each DMA burst.

Offset: 0x0+ (n * 0x40) | Read/Write: R/W | Reset: 0x00000000 (0b00000xxxxx000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	ENB: Enables DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Generates interrupts when DMA block transfer completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Holds this Processor until DMA block transfer completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction. 0 = APB read to AHB write 1 = AHB read to APB write. 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel). 0 = Run for Multiple Block Transfers 1 = Run for One Block Transfer. 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers). 0 = Independent of DRQ request 1 = Link to DRQ source. 0 = DISABLE 1 = ENABLE
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = QSPI 6 = APBIF_CH4 7 = APBIF_CH5 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = APBIF_CH6 13 = APBIF_CH7 14 = APBIF_CH8 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = Reserved 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = Unused, reserved 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = APBIF_CH9 30 = I2C6 31 = Reserved

21.2.6.2 APBDMACHAN_CHANNEL_n_STA_0

APB-DMA-n Status Register

There are 32 APB DMA Status Registers, one per channel (n = 0 through 31).

Offset: 0x4 + (n * 0x40) │ Read/Write: R/W │ Reset: 0xXX000000 (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RO	X	BSY: Indicates whether DMA Channel Status is active or not 0 = Inactive 1 = Active 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT
28	RO	X	PING_PONG_STA: If dir = AHB_WRITE: 0 - Ping buffer transfer completed; 1 - Pong buffer transfer completed. If dir = AHB_READ: 0 - Pong buffer transfer completed; 1 - Ping buffer transfer completed; 0 = PING_INTR_STA 1 = PONG_INTR_STA
27	RO	X	DMA_ACTIVITY: Indicates current DMA channel is transferring data 0 = IDLE 1 = BUSY
26	RO	X	CHANNEL_PAUSE: Indicates the status of channel pause.

21.2.6.3 APBDMACHAN_CHANNEL_n_DMA_BYTE_STA_0

APB-DMA-n DMA Byte Status Register

There are 32 APB DMA Byte Status Registers, one per channel (n = 0 through 31).

Offset: 0x8+ (n * 0x40) │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

21.2.6.4 APBDMACHAN_CHANNEL_n_CSRE_0

APB-DMA-n Control-Extended Register

There are 32 APB DMA Control Extended Registers, one per channel (n = 0 through 31).

Offset: 0xc+ (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxx000000xxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CHANNEL_PAUSE: When enabled, pauses data transfers on the channel 0 = RESUME 1 = PAUSE

Bit	Reset	Description
19:14	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15 25 = APB_16 26 = APB_17 27 = APB_18 28 = APB_19 29 = APB_20 30 = APB_21 31 = APB_22 32 = APB_23 33 = APB_24 34 = APB_25 35 = APB_26 36 = APB_27 37 = APB_28 38 = APB_29 39 = APB_30 40 = APB_31

21.2.6.5 APBDMACHAN_CHANNEL_n_AHB_PTR_0

APB-DMA-n AHB Starting Address Pointer Register

There are 32 APB Starting Address Pointer Registers, one per channel (n = 0 through 31).

Offset: 0x10 + (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: Software writes to modify

21.2.6.6 APBDMACHAN_CHANNEL_n_AHB_SEQ_0

APB-DMA-n AHB Address Sequencer Register

There are 32 APB DMA AHB Address Sequencer Registers, one per channel (n = 0 through 31).

Offset: 0x14 + (x * 0x40) │ Read/Write: R/W │ Reset: 0x26000000 (0b00100110xxxx0000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = Send interrupt to COP 1 = CPU 0 = COP

Bit	Reset	Description
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width. 0 = 8-bit bus (RSVD). 1 = 16-bit bus (RSVD). 2 = 32-bit bus (default). 3 = 64 bit-Bus (RSVD). 4 = 128-bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled, the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE
26:24	0x6	AHB_BURST: AHB Burst Size DMA Burst Length (encoded). 4 = 1 Word (1x32bits). 5 = 4 Words (4x32 bits). 6 = 8 Words (8x32bits), default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (for Run-Multiple Mode with No Wrap Operations). 1 = Reload Base Address for 2X blocks (reload every other time). 0 = Reload Base Address for 1X blocks (default) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window. 0=No Wrap (default). 5=Wrap on 512 word window. 1=Wrap on 32 word window. 6=Wrap on 1024 word window. 2=Wrap on 64 word window. 7=Wrap on 2048 word window. 3=Wrap on 128 word window. 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

21.2.6.7 APBDMACHAN_CHANNEL_n_APB_PTR_0

APB-DMA-n APB Starting Address Pointer Register

There are 32 APB DMA APB Starting Address Pointer Registers, one per channel (n = 0 through 31).

Offset: 0x18 + (n * 0x40) │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus

21.2.6.8 APBDMACHAN_CHANNEL_n_APB_SEQ_0

APB-DMA-n APB Address Sequencer Assignments

There are 32 APB DMA APB Address Sequencer Registers, one per channel (n = 0 through 31).

Offset: 0x1c + (n * 0x40) | Read/Write: R/W | Reset: 0x20010000 (0bx0100xxxxxxxx001xxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8-bit bus. 1 = 16-bit bus. 2 = 32-bit bus (default) 3 = 64-bit bus. (RSVD). 4 = 128-bit bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled, the data going to the APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window. 0 = No Wrap. 1 = Wrap on 1 Word Window (default). 2 = Wrap on 2 Word Window. 3 = Wrap on 4 Word Window. 4 = Wrap on 8 Word Window. 5 = Wrap on 16 Word Window. 6 = Wrap on 32 Word Window. 7 = Wrap on 64 Word Windows (rsvd). 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

21.2.6.9 APBDMACHAN_CHANNEL_n_WCOUNT_0

APB-DMA-n Word Count Register

There are 32 APB DMA Word Count Registers, one per channel (n = 0 through 31).

Offset: 0x20 + (n * 0x40) | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000000000000xx)

Bit	Reset	Description
29:2	0x0	WCOUNT: Number of 32-bit word cycles.

21.2.6.10 APBDMACHAN_CHANNEL_n_WORD_TRANSFER_0

APB-DMA-n Word Transfer Register

There are 32 APB DMA Word Transfer Registers, one per channel (n = 0 through 31).

Offset: 0x24 + (n * 0x40) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:2	X	COUNT: APB-Current 32-bit Word Cycles. Flags set/cleared by hardware.

CHAPTER 22: USB COMPLEX

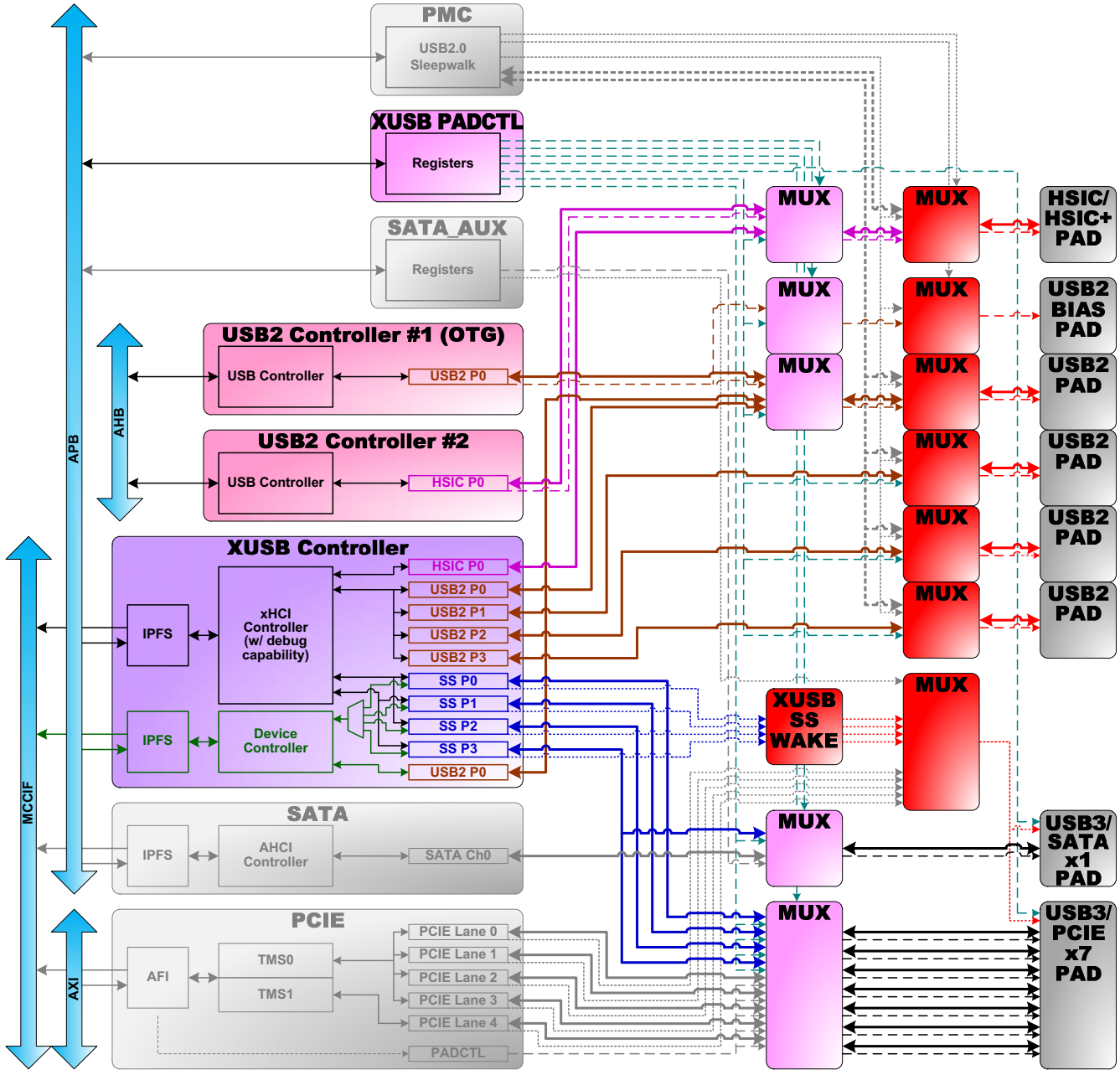
Note: *The ULPI functionality described in this section has been deprecated from Tegra® X1 devices, and is not supported by NVIDIA.*

22.1 USB Overview

The Tegra X1 device has two USB 2.0 controllers (named USB_OTG and USB2) and one XUSB 3.0 (xHCI/Device) controller (named XUSB), controlling a total of eight exposed ports and one internal only port shared among them. There are a total of 4 USB2 ports and 4 USB3 ports, and 1 HSIC port. Of the eight ports, one can be USB3.0 OTG or USB2.0 OTG while the rest are Host Mode only ports.

The following diagram shows the relationship between the different USB controllers and interfaces. Note that some USB ports are shared between the USB 2.0 and XUSB controllers. The HSIC pad supports one port. The APB slave provides the programmability of the owner of the pad to be a USB2 or XUSB controller.

Figure 51: Tegra X1 USB Controllers and Interfaces



The XUSB Pad Control logic and the USB2 controllers are allocated under the ungated partition of Vaux_soc power domain, where the XUSB Host Mode and Device Mode, PCIe, and SATA are allocated under power-gate-able partitions that are separately power gated.

The XUSB Pad Control logic is a stand-alone APB slave that accepts downstream requests for programming the Pad MUX and Pad specific parameters. The XUSB Pad Control logic also provides the programmability of the capabilities of the individual ports of XUSB, where PCIe and SATA each have their own per-port controls for the UPHY, and USB2 controllers each have their own per-port controls for the USB2.0 pads.

Below is a summary of the key components in the USB complex:

- USB_OTG and UTMIP: This is the primary USB port that supports OTG. USB recovery is also supported on this interface.
- USB2 and UHSIC: This interface allows connection of an HSIC peripheral/host on the PCB board.
- XUSB and UTMIP3. XUSB can support multiple interfaces at the same time.

- XUSB and UTMIP2. XUSB can support multiple interfaces at the same time.
- XUSB and UTMIP1. XUSB can support multiple interfaces at the same time.
- XUSB and UTMIP. XUSB can support multiple interfaces at the same time.
- XUSB and UHSIC. XUSB can support multiple interfaces at the same time.
- XUSB and SS. XUSB can support multiple interfaces at the same time, up to 4 Host Modes or 3 Host Modes and one OTG mode.

USB_OTG supports 16 bidirectional endpoints in Device Mode, where USB2 should never be advised or programmed to support Device Mode. Endpoint 0 of USB_OTG is always a bidirectional control endpoint. All other endpoints can be programmed as one of Bulk, Interrupt, Isochronous, or Control type in either direction. Control endpoints are always bidirectional and hence to program an endpoint as control type, both IN and OUT directions need to be programmed to control type. Conversely, if an endpoint has either IN or OUT direction programmed to be non-control type, the other direction also needs to be programmed to be non-control type. The maximum packet size supported on any endpoint is 1024 bytes in high-speed mode, for both Device and Host Modes.

VBus and ID detection are supported via an external PMIC. The input for VBUS detection status connects via dedicated sideband signals from the VGPIO. In addition, the APB slave should implement local software override bits to allow software to directly program the VBUS status to the APB Slave that bypasses the VGPIO.

22.2 USB 2.0 Controllers and Interfaces

The USB 2.0 controllers in the Tegra X1 mobile processors provide mechanisms for the processor to communicate with a PC as a peripheral or to communicate with USB 2.0 peripherals, such as keyboards, mouse devices, cameras, or storage devices, as a host using regular USB 2.0 ports. The USB 2.0 controllers also provide mechanisms for Tegra X1 devices to communicate with an on-board baseband controller with HSIC or regular UTMIP interfaces. Each Tegra X1 device provides two USB 2.0 controllers with five USB interfaces: four regular USB ports and one HSIC (see Figure 51).

The Tegra X1 device supports peripheral functionalities with USB 2.0 controller #1, which implements interrupt mechanisms to support device and host OTG operations with the regular USB 2.0 port. Tegra X1 devices support battery charging charger detection capabilities and associated interrupt mechanisms that are compliant with USB Battery Charging specification version 1.2 with USB 2.0 controller #1. For battery charging, software programming sequences are required to support the operations in response to hardware notifications from USB 2.0 controller #1.

Both USB 2.0 controllers support the EHCI programming model for scheduling transactions and interface managements as hosts. Each USB 2.0 controller contains an integrated transaction translator hub to support USB 1.1 transactions when connected to a USB 1.0 peripheral. USB 2.0 controller #1 supports EHCI-like programming models for scheduling responses to host requests.

Tegra X1 USB 2.0 controllers support L1 and L2 (suspend) link power managements. Tegra X1 USB 2.0 controllers support remote wakeup, wake on connect, wake on disconnect, and wake on overcurrent in all Tegra X1 power states, including deep sleep mode.

22.2.1 USB 2.0 Controller #1 (USB_OTG)

USB 2.0 Controller #1 only connects to USB2 port 0 (USB2 P0), which is the primary USB 2.0 port on Tegra X1 devices. USB 2.0 Controller #1 shares the same USB2 P0 pins with the USB 3.0 xHCI controller. The pinmux must be programmed prior to USB 2.0 Controller #1 using USB2 P0.

USB 2.0 Controller #1 supports both USB 2.0 device and USB 2.0 host operations. USB recovery is supported only with USB 2.0 Controller #1 and USB2 P0.

Wake-up from deep sleep mode is also supported on VBUS and accessory detect (ACCx_DETECT) pins.

22.2.2 USB 2.0 Controller #2

USB 2.0 Controller #2 must always be configured to Host mode, and it can be configured to connect to HSIC Port #1.

HSIC

USB 2.0 Controller #2 can be configured to use HSIC interface #1 to allow connection of an on-board peripheral supporting an HSIC interface to the Tegra X1 device. HSIC can support baseband controllers that support an HSIC interface.

22.2.3 Interface Restrictions

USB controller #2 is exclusively used for HSIC.

22.2.4 AHB Interface

Both USB 2.0 controllers adapt the AHB interface as masters and slaves for communicating with the other Tegra X1 devices. The following table lists the AHB interface numbers for each controller.

Table 105: AHB Master and Slave Numbers

Controller	AHB Master Number	AHB Slave Number
USB 2.0 Controller #1	6	6
USB 2.0 Controller #2	18	9

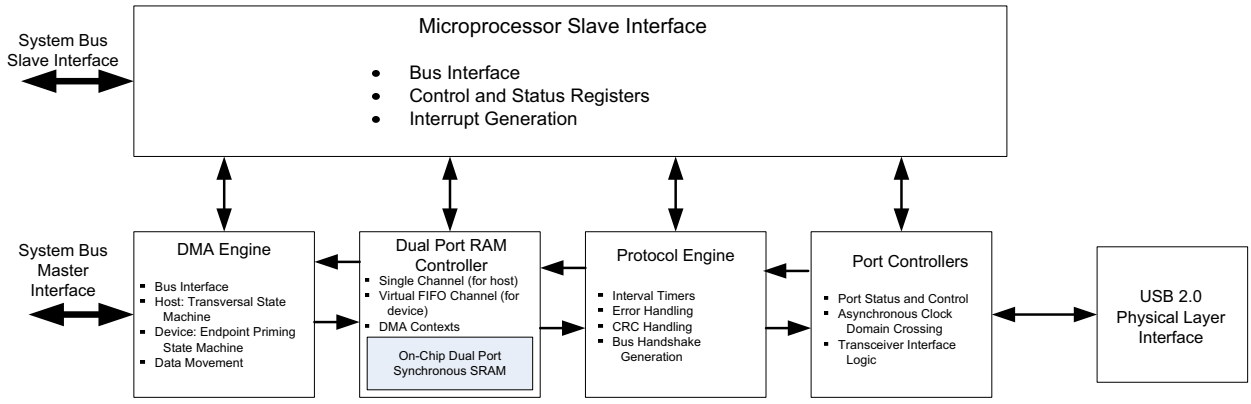
22.3 USB 2.0 Programming Interfaces

Both USB 2.0 controllers offer the same functionalities and may be configured to either host or device. However, due to interface restrictions, only USB 2.0 controller #1 should be configured to Device Mode, while USB 2.0 controller #2 must always be configured to Host Mode.

- Host controller registers and data structures are implemented as standard EHCI programming interface.
- Host controller supports USB 1.1 Full and Low speed devices via integrated transaction translator hub through EHCI standard data structures, instead of utilizing companion USB 1.1 host controller.
- Device controller registers and data structures are implemented as extensions to the EHCI programmers interface.

Figure 52 illustrates the control and datapaths block diagram of USB 2.0 controllers.

Figure 52: USB 2.0 Controller Block Diagram



22.3.1 Endpoint Capabilities

USB 2.0 controller #1 supports 16 IN and 16 OUT endpoints in Device Mode. Endpoint 0 is always a bidirectional control endpoint. All other endpoints can be configured as Bulk, Interrupt, Isochronous or Control endpoints in either direction. Control endpoints are always bidirectional. Hence for a control endpoint, the endpoints in both directions must be configured to be control endpoint. Conversely, if a particular endpoint number has either IN or OUT direction programmed to be a non-control endpoint, the other direction must also be configured as a non-control endpoint.

The maximum packet size supported on any endpoint is 1024 bytes in high-speed mode.

22.4 XUSB Controller

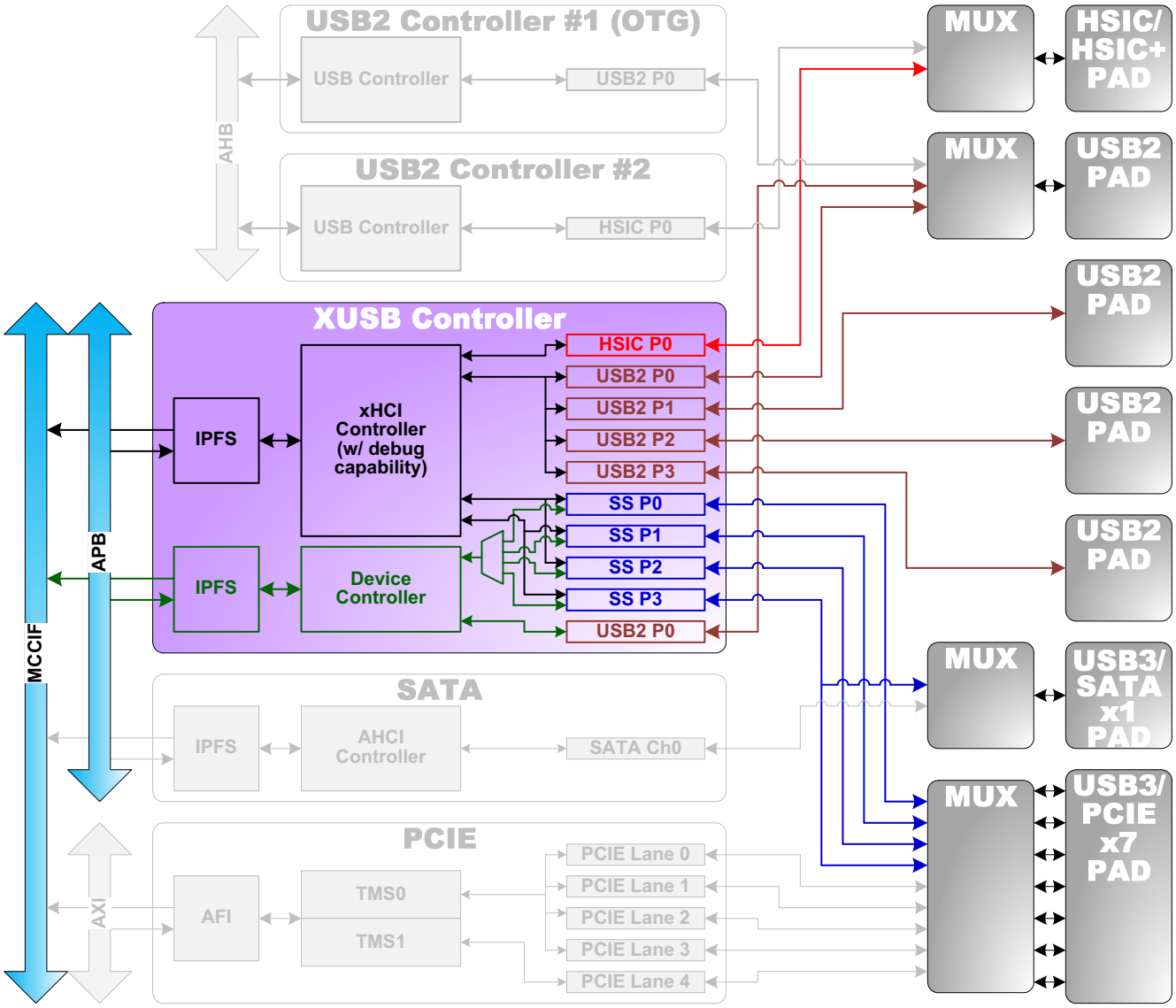
The Tegra X1 device provides 1 XUSB controller that can support up to 4 regular USB 3.0 ports and their companion regular USB 2.0 ports, or up to 4 standalone USB 2.0 ports (one of the four could be OTG). The XUSB controller provides mechanisms to communicate with USB 2.0 peripherals, such as keyboards, mouse devices, and card readers. The USB 3.0 controller also provides mechanisms to communicate with USB 3.0 peripherals, such as cameras and storage devices, as a host using regular USB 3.0 ports.

The XUSB controller supports xHCI Debug Capability. The xHCI Debug Capability supports only super speed link with one control, one bulk IN and one bulk OUT endpoints.

The XUSB controller supports Device Mode allowing the mobile platform to be accessed from a host device. The Device Mode supports both high/full speed and super speed with up to 15 IN and 15 OUT endpoints, where a control endpoint consists of one bidirectional endpoint. The endpoints can be configured by the driver to support transfer types of different device classes such as modem, storage, or input devices.

The following figure illustrates the connection between the XUSB controller and USB interfaces in Tegra X1 devices, where each USB 3.0 receptacle must encompass 1 USB 3.0 port and 1 USB 2.0 port from the XUSB controller. The XUSB Controller shares the same USB 2.0 Port #0 pins with USB 2.0 controller #1. The pinmux must be programmed prior to XUSB Controller using USB 2.0 Port #0.

Figure 53: Tegra X1 XUSB Controller and Interfaces



The XUSB controllers support the xHCI programming model for scheduling transactions and interface managements as a host that natively supports USB 3.0, USB 2.0, and USB 1.1 transactions with its USB 3.0 and USB 2.0 interfaces.

The Tegra X1 XUSB controller supports USB 2.0 L1 and L2 (suspend) link power management and USB 3.0 U1, U2, and U3 (suspend) link power managements. The XUSB controller supports remote wakeup, wake on connect, wake on disconnect, and wake on overcurrent in all Tegra X1 power states, including deep sleep mode.

USB 2.0 Ports

Each USB 2.0 port operates in USB 2.0 High Speed mode when connecting directly to a USB 2.0 peripheral. Each USB 2.0 port operates in USB 1.1 Full and Low Speed modes when connecting directly to a USB 1.1 peripheral.

All USB 2.0 ports operating in High Speed mode share one High Speed Bus Instance, which means 480 Mb/s theoretical bandwidth is distributed across these ports. All USB 2.0 ports operating in Full or Low Speed modes share one Full/Low Speed Bus Instance, which means 12 Mb/s theoretical bandwidth is distributed across these ports.

USB 2.0 ports of the XUSB controller support software initiated L1 and L2 (suspend) link power management. USB 2.0 ports of the XUSB controller do not support hardware initiated L1 link power management.

USB 3.0 Ports

USB 3.0 ports only operate in USB 3.0 Super Speed mode. All USB 3.0 ports share one Super Speed Bus Instance, which means 5 Gb/s theoretical bandwidth is distributed across these ports.

USB 3.0 ports of the XUSB controller support hardware initiated U1 and U2 link power management as well as software initiate U3 (suspend) link power management.

Note: *For details on USB2.0 and USB3.0 wake host capability, refer to the document entitled eXtensible Host Controller Interface for Universal Serial Bus (xHCI) on the Intel website:* http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/extensible-host-controler-interface-usb-xhci.pdf

---

Falcon Microcontroller

The Tegra X1 XUSB controller integrated an NVIDIA Falcon microcontroller to perform the following tasks:

- Command ring processing
- Event ring management
- Endpoint scheduling
- Context save and restore

22.4.1 Interface Restrictions

A USB3.0 connector includes both USB2.0 and USB3.0 interface signals. The USB2.0 interface signals of a USB3.0 connector must be assigned to the USB3.0 controller for xHCI specification compliance.

Similarly, if the USB3.0 interface signals are used to connect to a peripheral on the system board, such as a USB3.0 hub, the USB2.0 signals connecting to that peripheral must be assigned to the USB3.0 controller.

Only USB2.0 Port 0 can be configured as a device port or an OTG port. Any one of the USB3.0 port can be exclusively paired with USB 2.0 Port 0 and operates as the USB 3.0 interface of the device port or OTG port. Either USB 2.0 or USB 3.0 interface would be active when connected to a host port.

22.4.2 Host and Memory Access Interfaces

The XUSB controller adapts the APB interface as its host interface for configuration and memory-mapped I/O register accesses. The XUSB controller adapts direct memory as its memory interface for direct memory accesses.

22.5 XUSB Device Mode

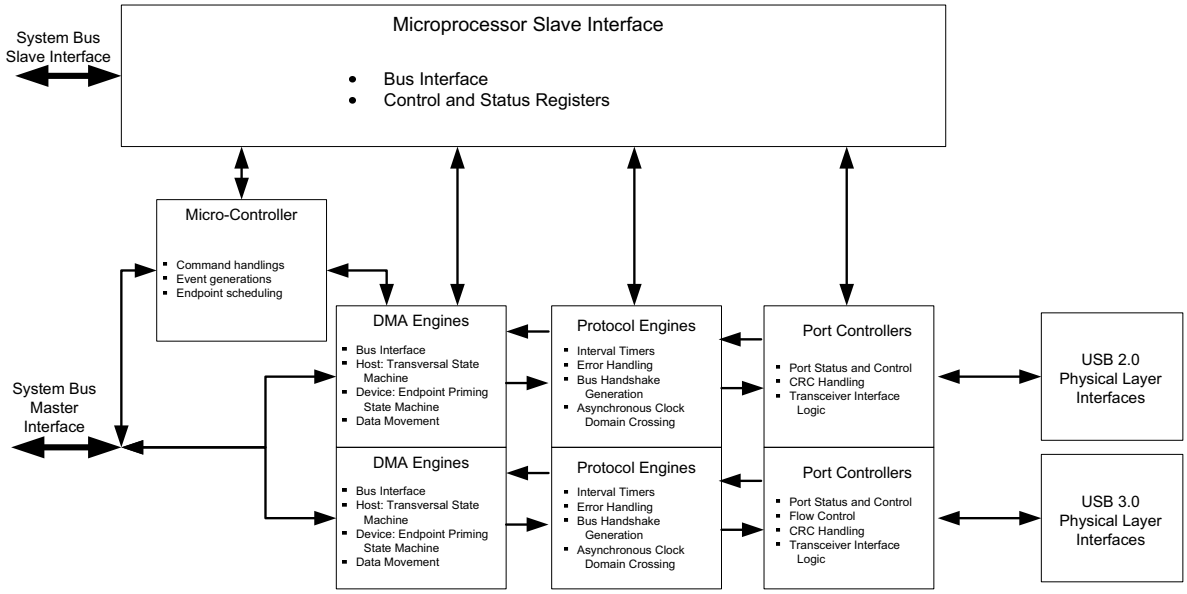
The xHCI host controller supports Debug Capability to satisfy the WHQL requirement of Microsoft. The xHCI Debug Capability supports only super speed link with one control, one bulk IN and on bulk OUT endpoints. It is expected that Microsoft will provide the drivers for both the host system and the system-under-debug to allow access to the system-under-debug via its xHCI Debug Capability.

The Device Mode allows the mobile platform to be accessed from a host device. The Device Mode supports both high/full speed and super speed with up to 15 IN and 15 OUT endpoints, where a control endpoint consists of one bidirectional endpoint. The driver of the Device Mode is expected to be developed in-house initially. The endpoints can be configured by the driver to support transfer types of different device classes such as modem, storage, or input devices.

22.6 XUSB Programming Interface

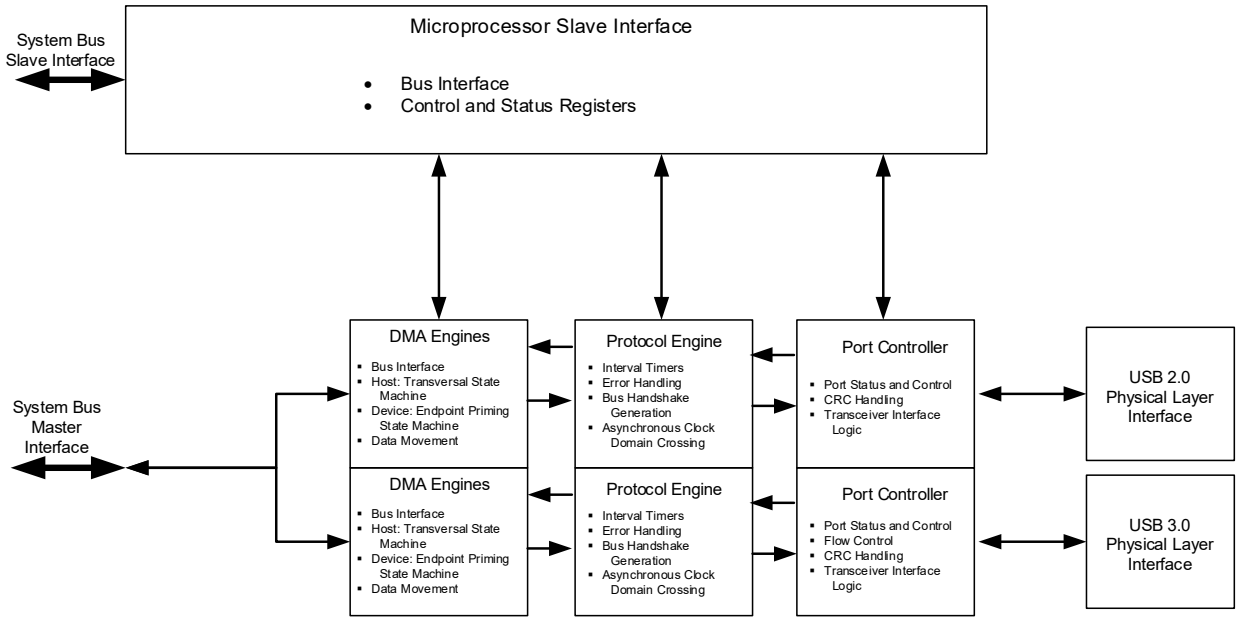
The XUSB Controller supports host functionality with host controller registers and data structures implemented as standard xHCI programming interface. The following figure illustrates the control and datapaths block diagram of the XUSB host controller.

Figure 54: XUSB Host Controller Block Diagram



The XUSB Controller supports device functionality with device controller registers and data structures developed to resemble xHCI programming interface. The following figure illustrates the control and datapaths block diagram of the XUSB device controller.

Figure 55: XUSB Device Controller Block Diagram

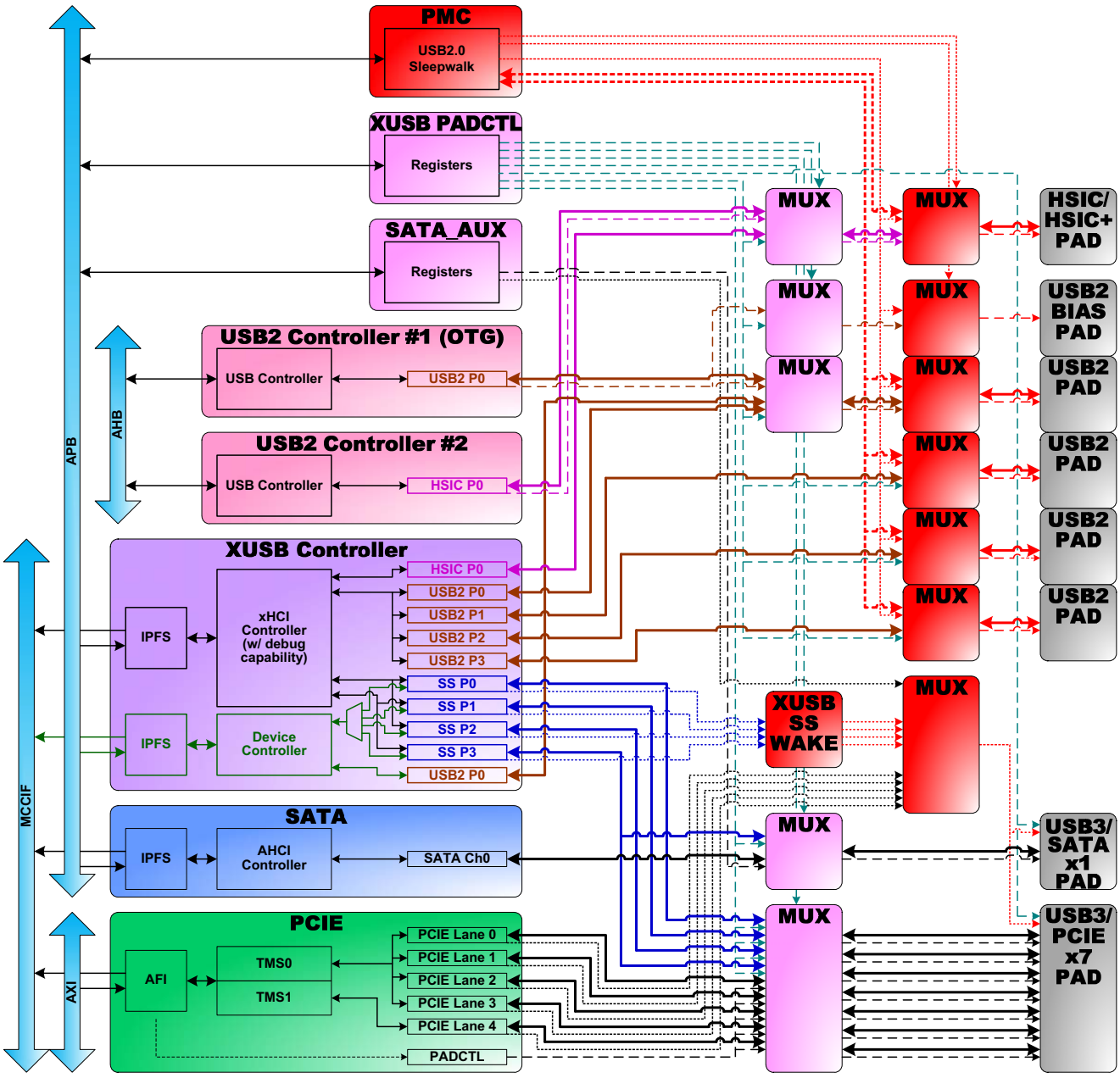


22.7 USB PADCTL

The XUSB Pad Control logic and the Synopsys USB2 controllers are allocated under the ungated partition of Vaux_soc power domain, where the XUSB Host Mode and Device Mode, PCIe, and SATA are allocated under power-gate-able partitions that are separately power gated.

The XUSB Pad Control logic is a stand-along APB slave that accepts downstream requests for programming the Pad MUX and Pad specific parameters. The XUSB Pad Control logic also provides the programmability of the capabilities of the individual ports of XUSB, where PCIe and SATA each have their own per-port controls for the UPHY, and USB2 controllers each have their own per-port controls for the USB2.0 pads.

Figure 56: Tegra X1 USB PADCTL and Interfaces



The following table lists the supported PCIe, XUSB, and SATA use cases in the Tegra X1 processor, where a 7-lane pad is shared between PCIe and XUSB's SuperSpeed ports and a 1-lane pad is shared with XUSB's SuperSpeed port and SATA.

Use Case	Lane P0	Lane P1	Lane P2	Lane P3	Lane P4	Lane P5	Lane P6	Lane S0
1A	PCIE x1 (Controller 1)	PCIE x2 (Controller 0)		USB SS (Port 2)	USB SS (Port 3)	USB SS (Port 1)	USB SS (Port 0)	SATA
1B	PCIE x1 (Controller 1)	PCIE x2 (Controller 0)		USB SS (Port 2)		USB SS (Port 1)	USB SS (Port 0)	USB SS (Port 3)
2A	PCIE x1 (Controller 1)	PCIE x4 (Controller 0)				USB SS (Port 1)	USB SS (Port 0)	SATA
2B	PCIE x1 (Controller 1)	PCIE x4 (Controller 0)				USB SS (Port 1)	USB SS (Port 0)	USB SS (Port 3)
3	USB SS (Port 2)	PCIE x2 (Controller 0)			USB SS (Port 3)	USB SS (Port 1)	USB SS (Port 0)	SATA

Use Case	Lane P0	Lane P1	Lane P2	Lane P3	Lane P4	Lane P5	Lane P6	Lane S0
4A	USB SS (Port 2)	PCIE x4 (Controller 0)				USB SS (Port 1)	USB SS (Port 0)	SATA
4B	USB SS (Port 2)	PCIE x4 (Controller 0)				USB SS (Port 1)	USB SS (Port 0)	USB SS (Port 3)

When set as a x4 controller, PCIe Controller 0 can further support the following modes, with lane 0 of the device connecting to either Lane P1 or Lane P4.

Use Case	Lane P0	Lane P1	Lane P2	Lane P3	Lane P4	Lane P5	Lane P6	Lane S0
		PCIE x4 (Controller 0)						
		PCIE x2 (Controller 0)						
				PCIE x2 (Controller 0)				
		PCIE x1 (Controller 0)						
					PCIE x1 (Controller 0)			

When set as an x2 controller, PCIe Controller 0 can further support the following modes, with lane 0 of the device connecting to either Lane P1 or Lane P2.

Use Case	Lane P0	Lane P1	Lane P2	Lane P3	Lane P4	Lane P5	Lane P6	Lane S0
		PCIE x2 (Controller 0)						
		PCIE x1 (Controller 0)						
			PCIE x1 (Controller 0)					

22.7.1 USB PADCTL Features

- Stand-alone APB slave
  - MMIO registers implemented under a stand-alone APB slave
  - MMIO space outside of xHCI host controller and Device Mode
- XUSB Port Control
  - Global configuration for XUSB ports
- Pad Control and Pad MUX
  - Pad ownership assignments for the PCIe pad, the SATA pad, and all USB 2.0 interface pads
  - Global pad programming outside of transaction protocols for the PCIe pad and the SATA pad
- VGPIO
  - VBUS and ID assertion/detection status via dedicated sideband signals from VGPIO
  - Software overrides allow software to directly update status from accessing PMIC
- Battery Charging
  - Supports standard and charging downstream port identification control when programmed as host/downstream ports
  - Supports charging port detection reporting as when programmed device/upstream ports
- On-the-Go
  - SuperSpeed and HighSpeed/FullSpeed device only – LowSpeed not supported
  - Supports OTG role swapping with SuperSpeed warm reset (RSP) and HighSpeed host negotiation protocol (HNP)

- Supports Session request protocol (SRP)
- Does not support attach detection protocol (ADP)

22.7.2 APB Interface

USB PADCTL adapts the APB interface as its host interface for configuration and memory-mapped I/O register accesses.

22.8 Programming Guidelines

22.8.1 USB_OTG Programming Guidelines

22.8.1.1 USB_OTG Programming

Clock Initialization

After power-on-reset, the USB clocks are disabled. Software can enable the USB clocks by setting CLK_ENB_USBD in the CLK_OUT_ENB_L register to ENABLE. Also, the USB controller stays in reset. Software should bring it out of reset by first setting SWR_USBD_RST in the RST_DEVICES_L register to ENABLE and then setting it to DISABLE.

Some parameters in UTMIP need to be programmed before the UTMIP can be brought out of reset. These parameters need to be programmed every time the USB controller is reset: the SWR_USBD_RST in RST_DEVICES_L register and RST in the USB2D_USBCMD register.

After UTMIP is reset, the PHY clock takes some time to come up, so software must wait until the USB_PHY_CLK_VALID bit in the USB_SUSP_CTRL register becomes valid. This bit, when set to 1, also generates an interrupt if RSM_IE is set in the USB_SUSP_CTRL register.

PHY Clock Control

The PHY clock can be turned off using the SUSP_SET bit in the USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

To turn on the PHY clock, software should write to SUSP_CLR in the USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

The current suspend status of the PHY can be checked by the USB_PHY_CLK_VALID bit of the USB_SUSP_CTRL register. If this bit is 1, the PHY clock is turned on; otherwise it is off.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

There is an interrupt generated whenever the PHY clock is turned on which can be checked by checking that the value of USB_PHY_CLK_VALID in the USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the RSM_IE bit in USB_SUSP_CTRL to 1/0.

Notes:

- *The PLLU_ENABLE bit in the PLLU_BASE register should be always set to ENABLE. All parameters for PLLU in the PLLU_BASE and PLLU_MISC registers should be set according to the oscillator frequency used. Not doing that will put the USB PHY PLL in a free-running mode, and can result in undesirable side effects.*
- *When the USB PHY is suspended, the PCLK should not be stopped. It can be reduced to 32 KHz, however. If the system clock is stopped, then it might not be possible to bring up the system on a wakeup event described below.*

There are two cases to consider for controlling the PHY clocks depending on whether the Tegra X1 device is in device or Host Mode.

Device Mode

The PHY clock can be turned off in two cases:

- When the USB cable is not connected, the VBUS signal is 0. In this case, software needs to turn on the PHY clock on cable insertion.
- When the USB cable is connected, the PHY clock can only be turned off when the SLI bit in the USB2D_USBSTS register is set to 1 when the USB Host puts the USB bus in suspend mode. This bit can be used to generate an interrupt if the SLE bit in the USB2D_USBINTR register is set to 1.

  In this case, software needs to set two bits in the USB_SUSP_CTRL register to turn on the PHY clock on a USB resume or a USB reset event from USB Host: WAKE_ON_DISCON_EN (Wake on Disconnect enable) and WK_RSM_EN (Wake on Resume enable).

  If WK_RSM_EN (bit 8) in the USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving a USB resume event from the USB Host. If WAKE_ON_DISCON_EN in the USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving a USB reset event from the USB Host or if the USB cable is disconnected. These bits should only be turned on after the PHY clock is turned off. The method described above can be used to interrupt the processor when the PHY clock is turned on. Also, turn off these bits when coming out of the suspend state.

Host Mode

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. After this, it can stop the PHY clock by using the method described above. The PHY clock resumes automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system in the suspend state by setting the SUSP bit in the USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into the suspend state. It can enable the WK_RSM_EN bit in the USB_SUSP_CTRL register and the WK_DS bit in the USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock is turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There is an interrupt associated with this as described above.

If the software wants to wake up the USB system, then it needs to turn on the PHY clock as described above.

22.8.1.2 LPM Support

Software should program the following register to set the duration that the host keeps generating resume signaling when the host initiated an exit from L1 based on system and device specific requirements:

CONTROLLER_USB2D_USBCMD_0.HIRD

The following software walk-around is required to extend the resume signaling if required.

1. Move pad out of Suspend

   IF_USB_SUSP_CTRL_0.USB_SUSP_CLR = SET (1)
2. Fake resume signaling

   UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE = 1

   UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE_SEL = 0xE
3. Wait the required time following the encoding of the HIRD value
4. Enable hardware to drive the resume signaling

   CONTROLLER_USB2D_PORTSC1_0.FPR = Resume driven on port (1)
5. Disable fake resume signaling as hardware takes over

   UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE = 0
6. Wait for hardware to finish driving resume signaling by checking the following bit:

   CONTROLLER_USB2D_PORTSC1_0.FPR = No resume driven on port (0)

7. Wait 50 µs
8. Enable hardware to start sending SOF and executing schedule

   CONTROLLER_USB2D_USBCMD_0.RS = RUN (1)

Software Initiated L1 Support

Software puts the bus into the L1 or Suspend (L2) state by setting the same SUSP bit in PORTSC1 register. The difference is to put the link to L1, software should first program the following registers:

CONTROLLER_USB2D_PORTSC1_0.DA = N, Device Address of the attached device

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. EPLPM = 0

CONTROLLER_USB2D_PORTSC1_0.SLP = Suspend using L1 (1)

Software should then set the following register bit to initiate L1 entry:

CONTROLLER_USB2D_PORTSC1_0.SUSP = 1

Software can check the following registers to ensure the link has entered L1:

CONTROLLER_USB2D_PORTSC1_0.SUSP = Port in suspend state (1)

CONTROLLER_USB2D_PORTSC1_0.SSTS = L1STATE_ENTERED (0)

If L1 cannot be entered successfully, the bus stays in the L0 state. Hardware will generate a port change interrupt when L1 entry fails. So software can check the following registers for the reason of the failed entry. Software can retry the L1 entry later if the reason is NYET_PERIPH, where the peripheral indicates it is not ready to put the bus into L1.

CONTROLLER_USB2D_PORTSC1_0.SUSP = Port not in suspend state (0)

CONTROLLER_USB2D_PORTSC1_0.SSTS = NYET_PERIPH (1), or

= L1STATE_NOT_SUPPORTED (2),

= PERIPH_NORESP_ERR (3),

When the bus is in the L1, software can set the following register at any time to bring the link back to the L0 state.

CONTROLLER_USB2D_PORTSC1_0.FPR = Resume driven on port (1)

Hardware Initiated L1 Support

Software can enable hardware to put the bus to L1 automatically. Software should first program the following registers, where *X* is the system specific idle threshold measured in number of SOFs with no activities in between:

CONTROLLER_USB2D_PORTSC1_0.DA = N, device address of the attached device

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. EPLPM = 0

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. LPMFRM = X

Software should then set the following register bit to enabled hardware initiated L1 entry:

CONTROLLER_USB2D_HOSTPC1_DEVLC_0. LPMX = 1

22.8.1.3 Streaming Mode

The USB controller supports streaming, where it starts sending DATA packets for OUT transactions when the controller has not finished fetching the entire data packet from system memory to its TX buffer. For IN transactions, the streaming mode allows data to be written to system memory before the entire data packet is received in its RX buffer. Streaming mode can be disabled with the following bit:

CONTROLLER_USB2D_USBMODE_0.SDIS

Enabling streaming mode allows lower latency and higher bandwidth of USB transfers but requires lower system memory latency.

Long system memory latency might cause TX buffer underrun/RX buffer overrun when streaming mode is enabled. Streaming mode should be kept enabled when USB controller is initialized and streaming mode should only be disabled if it is determined the system latency does cause the buffer overrun/underrun issues.

22.8.2 USB2 Programming Guidelines

The USB2 controller supports a UHSIC interface. All of these interfaces are directly controlled through USB2 register space.

22.8.2.1 Clock Initialization

After power-on-reset, the USB clocks are disabled. Software can enable the USB clocks by setting CLK_ENB_USB2 in CLK_OUT_ENB_L register to ENABLE.

Also, USB stays in reset. So software should bring it out of reset by first setting the SWR_USB2_RST in RST_DEVICES_L register to ENABLE and then setting it to DISABLE.

After the clocks for USB2 are up, software can program any register for USB2 required for proper configuration.

PLLU outputs a 480 MHz clock (FO_UHSIC) for the UHSIC interface. If using any of these interfaces, PLLU should be programmed appropriately.

Any time after USB2 is reset, or the UHSIC PHY comes out of suspend, software needs to wait until ULPI/UHSIC PHY clock comes up. It can do that by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. If this bit is 1, PHY clocks are up; otherwise, they are not. There are interrupts associated with this bit: if USB_PHY_CLK_VALID_INT_ENB is set to ENABLE, USB_PHY_CLK_VALID_INT_STS will be set to SET whenever the PHY clock becomes valid. Software can write a 1 to USB_PHY_CLK_VALID_INT_STS to clear the interrupt status.

Before doing anything useful on USB2, one of its interfaces needs to be selected and configured.

Selection of UHSIC Interface

1. The UHSIC I/O pad is in the power-down state at power-on. Bring it out of power-down mode by setting the PD_BG, PD_TX, PD_RX, PD_ZI and RPD_DATA, RPD_STROBE fields in the UHSIC_PADS_CFG1 register to 0.
2. Bring the tracking circuit out of power-down mode by clearing the PD_TRK bit.
3. Add 25 µs delay to allow calibration to complete.
4. Tracking circuit can be powered down now by setting PD_TRK=1.
5. Hold UHSIC in reset by writing 1 to the UHSIC_RESET field in the IF_USB_SUSP_CTRL register.
6. Program the PLLU parameters to enable the UHSIC PLLU clock.
7. Select the UHSIC interface by writing 1 to the UHSIC_PHY_ENB field in the IF_USB_SUSP_CTRL register.
8. Program the configuration parameters for UHSIC.
9. Release reset to the UHSIC by writing 0 to the UHSIC_RESET field in the IF_USB_SUSP_CTRL register.
10. Set the CM field in the USB2D_USBMODE register to HOST mode.
11. Program the USB2 controller to use the UHSIC PHY by writing 0 to the PTS field in the CONTROLLER_1_USB2D_HOSTPC1_DEVLC register.
12. Program the UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE field in the UHSIC_TX_CFG0 register to “1”.
13. Wait until the PHY clock comes up by checking the USB_PHY_CLK_VALID bit in the IF_USB_SUSP_CTRL register. An interrupt can be generated as explained above.

22.8.2.2 PHY Clock Control

The PHY clock can be turned off using the PHCD bit in the USB_PORTSC1 register. Note: This bit does not need to be pulsed.

To turn on the PHY clock, software should write to USB_SUSP_CLR in USB2_IF_USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

Whenever the PHY is placed in the suspend mode, the PHY clock is turned off and USB_PHY_CLK_VALID in the USB2_IF_USB_SUSP_CTRL register is set to 0. Software should make sure that the PHY clock shuts down by polling this bit whenever it places the PHY into suspend mode.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in the USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

An interrupt is generated whenever the PHY clock is turned on, which can be checked by checking the value of USB_PHY_CLK_VALID in the USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the USB_PHY_CLK_VALID_INT_ENB bit in the USB_SUSP_CTRL register to 1/0.

To clear the USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS interrupts, software can write a 1 to corresponding bits.

Generally, whenever the PHY is woken up from the suspend mode, first a wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by a PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when PHY clock starts up.

22.8.2.3 Bus Signaling Procedure Changes for the UHSIC Interface

To support the UHSIC interface, some changes are required in the bus signaling procedure for USB2 controller (connect and bus reset sequences) as described in the subsections below:

Host Mode - Bus Connect Sequence

The normal connect sequence assumes that port power has been enabled on a downstream port and the device has signaled a “Connect” on the bus. This will generate a port change interrupt and the software takes action from here. Standard EHCI connect interrupt can be used to detect the device connect.

Host Mode: Bus Reset Sequence

This is a reconnect procedure without physically detaching the USB cable. This operation is demanded by software whenever necessary as it is the case when an unrecoverable anomaly takes place. It would then bring the bus, the port, and the device into a well-known state.

1. Check if PortReset (the PR bit in the USB2D_PORTSC1 register) is NOT active. If by any means a bus reset is already taking place, it must be stopped before proceeding. This is done by clearing the bit and polling until it goes LOW.
2. Enable the “PortReset” bit - Start the HSIC bus reset on a downstream port by activating the PR bit of the USB2D_PORTSC1 register.
3. Wait until the end of the reset. This wait time is defined by the USB 2.0 specification and has the effect of holding the reset signaling on the HSIC bus.
4. Clear the PR bit.Stop the reset signaling by clearing the respective port reset bit.
5. Poll until the reset bit goes LOW to guarantee that the reset bit is effectively cleared;
6. Poll until LineState is DPlus.This confirms that the reset signaling has really finished on the HSIC bus and that the USB2 controller is ready to connect again.
7. Proceed with the “HOST Bus Connect Sequence” as described above. From here on, the procedure is the same as with an initial connect, instructing the USB2 controller to enter HS directly, skipping speed negotiation.
8. Standard EHCI reset interrupt can be used in Tegra X1 HSIC because native HSIC support is built-in.

22.8.3 UTMIP Programming Guidelines

A Tegra X1 device has a total of 4 UTMIP interfaces and four cabled USB ports. These ports are not part of a multiport host configuration. They are completely independent ports and can be configured in any combination of device or host. Internally, these ports correspond to USB controller 2.0, and any three XUSB controllers. UTMIP Port 0 is controlled only by the USB 2.0 controller. Externally, these are USB ports 0, 1, 2, and 3. All UTMIP PHY interfaces are identical instances of the same design. All four controllers use latest edition of USB core controller.

22.8.3.1 Holding USB in Reset

It is important that most static configuration of the UTMIP (and USB) be done while the unit is held in reset. For example, holding reset ensures that PLL_U is disabled and can be reconfigured. It also ensures that transactions are not occurring on the USB interface. Holding reset in both a controller and its corresponding PHY can be achieved by using the RST bit of the USB2D_USBCMD register.

22.8.3.2 PLL_U Programming

The USB ports use a cascaded PLL scheme to guarantee a high clock quality. First there is a PLL_U, which is the source clock for both the USB PLLs that are dedicated to ports. PLL_U (of type CLKPLL960_USB) produces reference clocks for all forms of USB (UTMIP, UHSIC). The table below describes the parameter settings that are dependent on the crystal clock reference frequency. The parameters are specified in decimal notation.

Table 106: 480 MHz PLL_U Output Frequency Fo = (Fi·N)÷(M· 2) With VCO_FREQ=0

Fi (MHz)	12.0 MHz	13.0 MHz	16.8 MHz	19. 2MHz	26.0 MHz	38.4 MHz	48.0 MHz
N (PLLU_DIVN)	960	960	400	200	960	200	960
M (PLLU_DIVM)	12	13	7	4	26	4	12

These parameters can be found in the PLLU_BASE register. The PLLU_OVERRIDE bit should be set to 0. PLLU_VCO_FREQ parameter must also be set to 0.

> ***Note:*** *PLL_U must be properly configured in the Boot ROM. DO NOT change the parameters of PLL_U while the unit is running. The same applies to the USB_PHY_PLL.*

22.8.3.3 PLL_U and USB_PHY_PLL Automatic Startup Times

The PLL_U and USB_PHY_PLL automatic startup times are used in reset and suspend modes to kick start the PLLs. Software should not manually force the PLL up and down because it cannot do so rapidly enough to meet the protocol. The following table lists the start times, which must be set up in the Boot ROM.

Table 107: PLL Automated Start Times (Must be Set Up in the Boot ROM)

Crystal Frequency	12.0 MHz	38.4 MHz
PLLU_ENABLE_DELAY_COUNT	2	5
PLLU_STABLE_COUNT	47	150
PLL_ACTIVE_DLY_COUNT	8	24
XTAL_FREQ_COUNT	118	375

PLLU_ENABLE_DLY_COUNT should be set for at least 1 microsecond, PLLU_STABLE_COUNT should be set for at least 1 ms, the PLL_ACTIVE_DLY_COUNT should be at least 10 us, and the XTAL_FREQ_COUNT set for about 2.5 ms.

22.8.3.4 Fuse Programming

The Tegra X1 device incorporates fuses to account for process variation in high speed data eye swing. The high speed data eye is controlled through the SETUP[6:0] transceiver pad parameter of the USB control registers. To set this parameter via the fuses, set the UTMIP_SPARE_CFG0[3] bit on all USB ports. This must be done in the Boot ROM. The default is to maintain

backward compatibility, which implies no fuses are used by default. The objective is to get the data eye swing as close as possible to 400mV without falling below the mark under typical operating conditions.

22.8.3.5 Programming the Tracking Length Time

The PD_TRK signal is used to power down the bias cell. After 25 microseconds of bias cell operation, the PD_TRK signal can be turned high to save power. This can be automated by programming a timing interval as given in the following table.

Table 108: 25 µs Timer on Bias Cell Tracking (for Set Up in the Boot ROM)

Crystal Frequency	12.0 MHz	38.4 MHz
UTMIP_BIAS_PAD_TRK_COUNT (UTMIP_BIAS_CFG1[7:3])	5	15

All values in the table are decimal. This parameter must be set in the Boot ROM while the crystal clock is stopped. To stop the crystal clock, disable the UTMIP_PHY_XTAL_CLOCKEN bit of register UTMIP_MISC_CFG1 and then re-enable it when done.

Because there is more than one UTMIP sharing the same bias cell, power down is done through a daisy chain, requiring power down across all ports.

22.8.3.6 Powering Down a USB Port Outside of Deep Power Down (DPD)

When a port is known to be disabled, there needs to be a mechanism to stop its power consumption for situations outside of DPD. This section addresses disabling a port while powered up.

Powering down of USB at times other than deep power down should be done by setting all specialized PLL and pad power down pins instead of using the global E_DPD pins of the USB analog components. It is safer to power down the cells through the traditional power down pins when the 3.3V and 1.2V supplies might still be on. This is achieved by setting the following pad controls from UTMIP register space. Before setting the power down bits; save previous registers values so that they may be restored. This type of power down should not be done in the Boot ROM.

Table 109: UTMIP Pad Controls

Pad Control	Analog Cell	Register Bit Settings
PD	Transceiver	UTMIP_FORCE_PD_POWER_DOWN=1
PD2	Transceiver	UTMIP_FORCE_PD2_POWER_DOWN=1
PD_ZI	Transceiver	UTMIP_FORCE_PDZI_POWER_DOWN=1
PD_DR	Transceiver	UTMIP_FORCE_PDDR_POWER_DOWN=1
PD_CHRP	Transceiver	UTMIP_FORCE_PDCHRP_POWER_DOWN=1
PD_DISC	Transceiver	UTMIP_FORCE_PDDISC_POWER_DOWN=1
PD_CHG	Transceiver	UTMIP_PD_CHG=1
PD	Bias	UTMIP_BIASPD=1
PD_TRK	Bias	UTMIP_FORCE_PDTRK_POWER_DOWN=1
OTG_PD	Bias	UTMIP_OTGPD=1
ID_PD	Bias	UTMIP_IDPD_SEL=1, UTMIP_IDPD_VAL=1
ENABLE	PHY PLL	UTMIP_FORCE_PLL_ENABLE_POWERDOWN=1
ACTIVE	PHY PLL	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN=1
PD_SAMP_A/C	PHY_PLL	UTMIP_FORCE_PD_SAMP_A/C_POWERDOWN=1
ENABLE	PLL U	UTMIP_FORCE_PLLU_POWERDOWN=1
UTMIP	PHY	UTMIP_PHY_XTAL_CLOCKEN=0

22.8.3.7 Extending Debouncer Period Length

USB debouncers provide a programmable debounce to alleviate the software burden. This is done with the UTMIP_DEBOUNCE_TIME_SCALE parameter located at UTMIP_BIAS_CFG1[13:8]. The default implies a timescale factor of

1. The timescale should not be programmed at Boot ROM time. The timescale register field incremented by 1 multiplies the debounce duration for all debouncers.

In the Boot ROM, the A debounce period should be set to 10 ms. This is dependent on the crystal frequency scale. The following table shows how it is programmed.

Table 110: 10 ms Timer on the A Debounce Period

Crystal Frequency	12.0 MHz	13.0 MHz	16.8 MHZ	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
UTMIP_BIAS_DEBOUNCE_A	30000	32500	42000	48000	65000	96000	120000

To program values of 96000 and 120000 for crystal clock frequencies of 38.4 and 48 MHz, respectively, these steps must be followed:

1. In the debounce register, program 48000 for a 38.4 MHz crystal clock or 60000 for a 48 MHz clock.
2. Set UTMIP_BIAS_DEBOUNCE_TIMESCALE to 1 by writing into the USB2_UTMIP_BIAS_CFG1[1] register field.

22.8.3.8 Deep Power Down Behavior

Tegra X1 devices' deep power down behavior for USB wake-up events is controlled in the PMC via registers USB_DEBOUNCE_DLY, USB_AO and the WAKE_MASK (USB_EVENT field) of the PMC unit. These control the wake-up events, their debounce duration, and their debounce length while powered down. USB port 0 has a possible event on ID or VBUS detection. These can be configured at startup time. These registers take over from the UTMIP registers when the chip is powered down. The programming depends on the context of the chip. For example, if USB port 0 is a dedicated host port then the VBUS wake-up event should be kept powered down. Conversely, a dedicated Device Mode port may not want to wake-up on the detection of an ID connector and can chose to power down ID.

22.8.3.9 Miscellaneous Boot ROM Fields

The UTMIP fields in the following table must be programmed at Boot ROM time. These should be set when the USB device is held in reset.

Field	Value
UTMIP_FS_PREAMBLE_J (UTMIP_TX_CFG0)	1
UTMIP_PD_CHG (UTMIP_BAT_CHRG_CFG0)	1
UTMIP_XCVR_LSBIAS_SEL (UTMIP_XCVR_CFG0)	0
UTMIP_SPARE_CFG0[3]	1
UTMIP_IDLE_WAIT (UTMIP_HSRX_CFG0)	17
UTMIP_ELASTIC_LIMIT (UTMIP_HSRX_CFG0)	16
UTMIP_HS_SYNC_START_DLY (UTMIP_HSRX_CFG1)	9

22.8.3.10 PHY Resets

Additional controls have been provided to stop the USB PHY (and its clocks) by issuing a reset to the PHY only (as opposed to a reset encompassing the controller and the PHY). This is akin to the Reset found in the UTMI+ standard. This is done in both controllers. See the UTMIP_RESET field of the USB_SUSP_CTRL register for more information.

22.8.3.11 Static Boot ROM Configuration for UTMIP

The boot ROM configuration for UTMIP should be done while the unit is held in Reset. For correct behavior, the following sequence MUST be followed:

1. Apply Reset to the USB controller (and UTMIP).
2. Run the crystal clock for approximately 5 microseconds while the UTMIP is in reset.
3. Stop the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN Low. This only stops the crystal clocks in the UTMIP units.

4. Program PLL_U as described in the “PLL_U Programming” section.
5. Program automatic PLL start times as described in the “PLL_U and USB_PHY_PLL Automatic Startup Times” section.
6. Remove power downs from USB_PHY_PLL ACTIVE/ENABLE/PD_SAMP_A/C and PLL_U.
7. Program the tracking duration as described in “Programming the Tracking Length Time.” Remove the power downs from PD (Bias) and disable PD_TRK. After 25 µs (tracking time), enable PD_TRK by writing 0 into UTMIP_FORCE_PDTRK_POWER_DOWN.
8. Once PD (Bias) is disabled, disable OTG_PD after 1 µs.
9. Remove powerdowns from ID_PD and VBUS_WAKEUP_PD. This can be removed earlier, as there is no timing or sequential relation with other signals.
10. Once the PLL_U, USB_PHY_PLL, and Bias pad are ready, remove PD (Transceiver). After 400 ns, the USB pad is ready for HS/FS/LS operation.
11. There are other powerdowns which can be removed if needed (as such not needed for Boot ROM). These are PD / PD_ZI / PD_DR / PD_CHRP / PD_DISC / PD_CHG.
12. Program the debouncer length time as described in “Extending Debouncer Period Length.”
13. Program various static parameters of the UTMIP as described in “Miscellaneous Boot ROM Fields.”
14. Restart the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN.
15. Resume any previous USB programming work that falls outside of UTMIP and release reset. From the UTMIP perspective it does not matter when reset is released, as long as it is after step 8. It will take about 3.5 ms before the 60 MHz clock appears once reset is released.

From a UTMIP perspective, ensure that all the cabled USB port PHYs are configured similarly in the Boot ROM.

22.8.3.12 EHCI Deviations

- Embedded Transaction Translator – Allows direct attachment of FS and LS devices in Host Mode without the need for a companion controller.
- Embedded design interface – This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

Programmable Physical Interface Behavior

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the HOSTPCx register providing a capability that is not defined by EHCI.

Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the PORTSCx register for a minimum duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed, there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. The basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a ‘1’ to the reset the device.
- Software shall write a ‘0’ to the reset the device after 10 ms.
- Driver needs to wait until port change interrupt is asserted and port reset is cleared

---

Note: *Should the EHCI host controller driver attempt to write a ‘0’ to the reset bit while a reset is in progress the write will be ignored and the reset will continue until completion.*

- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device is now operational and at this point the port speed has been determined.

Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non high-speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator (PSPD) has been added to HOSTPCx to provide the current operating speed of the port to the host controller driver.

Port Reset duration is 55 ms instead of 50 ms.

Port Suspend (PORTSC.Suspend) bit is set immediately after setting the bit. As per EHCI a 4 ms delay is expected for this bit to be set.

Tegra X1 devices also have the following cases that are not directly compliant to EHCI:

- USB controller/PHY interface initialization
- PHY Clock shutdown/wakeup procedure during suspend/resume/connect/disconnect
- PMC logic use for special low power modes during suspend/LP0

Notes:

- *UTMIPLL can be sourced from the osc_clk (38.4 MHz) or PLLU output (480 MHz). Default source is osc_clk. In cases, where PLLU is being used as a source for UTMIPLL reference clock, ensure that PLLU is enabled and locked before enabling UTMIPLL.*
- *PLLE can be sourced from the osc_clk (38.4 MHz) or PLL_REFE output (624 MHz). Default source is osc_clk. In cases, where PLL_REFE is being used as a source for PLLE reference clock, ensure that PLL_REFE is enabled and locked before enabling PLLE.*
- *Both Brick PLLs are sourced from PLLE (100 MHz) output. So, PLLE should always be enabled and locked first before enabling Brick PLLs.*
- *For PLLE and pex_pad PLL/sata_pad PLL, PLLE needs to be running when software is enabling pex_pad PLL or sata_pad PLL. The hardware sequencer cannot be enabled for PLLE (as it might disable PLLE) until software has enabled brick PLLs and subsequently has enabled the Brick PLL hardware sequencers as well.*
- *Boot loaders should perform the minimum PLL programming for access boot media. For example, if boot from PCIe is desired, boot loader should initialize PLLE and pex_pad PLL. In this case sata_pad PLL should be left uninitialized.*
- *Boot loader should not enable hardware power sequencer. Boot loader should enable the required PLLs in software controlled state.*
- *In order to hand off UPHY PLLs and PLLE cleanly to the next stage of boot loader or the main OS, (after boot loader completes access to boot media), boot loader should disable PLLs and assert resets to UPHY.*

22.8.4 Cold Boot with no Recovery Mode or Boot from USB

Step 1

Boot ROM enables USB2.0 related PLLs with software override.

1. Access the following CAR registers to measure the frequency of osc_clk.
   - CLK_RST_CONTROLLER_OSC_FREQ_DET_0[REF_CLK_WIN_CFG]

- CLK_RST_CONTROLLER_OSC_FREQ_DET_0[OSC_FREQ_DET_TRIG]
- CLK_RST_CONTROLLER_OSC_FREQ_DET_STATUS_0[OSC_FREQ_DET_BUSY]
- CLK_RST_CONTROLLER_OSC_FREQ_DET_STATUS_0[OSC_FREQ_DET_CNT]
- CLK_RST_CONTROLLER_OSC_CTRL_0[OSC_FREQ]

Set up the PLLU (enable in software):

2. Program these registers:
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_OVERRIDE] = 1 (Default)
   - CLK_RST_CONTROLLER_PLLU_MISC_0[PLLU_IDDQ]= 1
   - CLK_RST_CONTROLLER_PLLU_MISC_0[PLLU_IDDQ]= 0
   - CLK_RST_CONTROLLER_PLLU_OUTA_0[PLLU_OUT1_RSTN]=RESET_ENABLE (Default)
   - CLK_RST_CONTROLLER_PLLU_OUTA_0[PLLU_OUT2_RSTN]=RESET_ENABLE (Default)
3. Wait 5 µs.

   DIVM/DIVN should be programmed to generate 480 MHz VCOCLK and DIVP should be programmed to do divide-by-2
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_DIVM]
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_DIVN]
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_DIVP]
   - CLK_RST_CONTROLLER_PLLU_MISC_0[PLLU_KCP]
   - CLK_RST_CONTROLLER_PLLU_MISC_0[PLLU_KVCO]
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_ENABLE] = 1
4. Wait for LOCK:
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_LOCK]
5. Enable the Clocks:
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_CLKENABLE*] = 1
6. Enable Post dividers:
   - CLK_RST_CONTROLLER_PLLU_OUTA_0[PLLU_OUT1_RSTN]=RESET_DISABLE
   - CLK_RST_CONTROLLER_PLLU_OUTA_0[PLLU_OUT2_RSTN]=RESET_DISABLE
7. Wait 2 µs.

Set up the UTMIPLL (enable in software):

8. If (UTMIPLL Source == osc_clk) {
   - CLK_RST_CONTROLLER_UTMIP_PLL_CFG3_0[UTMIP_PLL_REF_SRC_SEL] = 0 (Default)
   - CLK_RST_CONTROLLER_UTMIP_PLL_CFG3_0[UTMIP_PLL_REF_DIS] = 0 (Default)
   - Skip to step 10.
9. Otherwise
   - CLK_RST_CONTROLLER_UTMIP_PLL_CFG3_0[UTMIP_PLL_REF_SRC_SEL] = 1
10. Program this register:
    - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE] = 0
11. Wait 10 µs.

12. Based on the reference clock source selected, MDIV/NDIV should be programmed to generate a 960 MHz VCOCLK. PLL has internal divide-by-2 (fixed) to generate a 480 MHz clock at the output.
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0[UTMIP_PLL_MDIV]
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0[UTMIP_PLL_NDIV]
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0[UTMIP_PLL_KCP]
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0[UTMIP_PLL_KVCO]
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0[UTMIP_PLL_VREG_CTRL]
13. USB2.0 Controller IP FSM configuration
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0[UTMIP_PHY_XTAL_CLOCKEN] = 1 (Default) [Gating for FSM input clock which is clk_m]
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0[UTMIP_PLL_ACTIVE_DLY_COUNT] = Don’t Care [No PLL Active control in UTMIPLL anymore]
14. If (UTMIPLL Source == osc_clk)
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0[UTMIP_PLLU_STABLE_COUNT] = 0x0.
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_PLLU_ENABLE_DLY_COUNT] = 0x0.
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLLU_POWERDOWN] = 1
    - Skip to step 16.
15. Otherwise:
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0[UTMIP_PLLU_STABLE_COUNT] = 0x3.

      [40 µs {8 µs} = (40*19.2/256)]
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_PLLU_ENABLE_DLY_COUNT] = 0x3.

      [1 µs = (1*19.2/8)]
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLLU_POWERDOWN] = 0
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLLU_POWERUP] = 0
16. Program this register:
    - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_XTAL_FREQ_COUNT] = 0x3.

      [40 µs {15 µs} = (40*19.2/256)] -> This should be total delay for the sequence, i.e., sum of both PLLs lock time + extra time for FSM states

Step 2

Boot ROM deasserts reset to XUSB PADCTL block.

1. Set the following CAR register bit to ‘0’ to deassert reset to XUSB Pad Control block.
    - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_PADCTL_RST]

Boot ROM deasserts reset to XUSB PADCTL block.

2. Set the following CAR register bit to ‘0’ to deassert reset to PCIE/XUSB PAD and SATA/XUSB PAD.

    -CLK_RST_CONTROLLER_RST_DEVICES_Y_0[SWR_PEX_USB_UPHY_RST]

    -CLK_RST_CONTROLLER_RST_DEVICES_Y

_0[SWR_SATA_USB_UPHY_RST]

Note: *All USB2.0 ports are assigned to USB2 controllers by default so no port assignment programming is required.*

Step 3

Boot ROM performs battery charging operations through USB2 Controller.

1. Set the following CAR register bits to '1' to enable the clocks to USB2 Controller, where only USB2 Controller #1 is required to be enabled for battery charging detection.
   - CLK_RST_CONTROLLER_CLK_ENB_L_SET_0[SET_CLK_ENB_USBD]
   - CLK_RST_CONTROLLER_CLK_ENB_H_SET_0[SET_CLK_ENB_USB2]
   - CLK_RST_CONTROLLER_CLK_ENB_H_SET_0[SET_CLK_ENB_USB3]
2. Set the following CAR register bits to '0' to deassert reset to USB2 Controller #1
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0[SWR_USBD_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0[SWR_USB2_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0[SWR_USB3_RST]
3. Set the following PMC register bits to '0' to deassert reset to USB2 Controller #1
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0[SWR_USBD_RST]
4. Set the following USBOTG register bits to follow the value programmed from FUSE to program the USB pad.
   - USB1_UTMIP_XCVR_CFG0_0[UTMIP_XCVR_SETUP_MSB,UTMIP_XCVR_SETUP] to {0, FUSE_USB_CALIB_0[5:0]}
   - USB1_UTMIP_XCVR_CFG1_0[UTMIP_XCVR_TERM_RANGE_ADJ] to FUSE_USB_CALIB_0[10:7]
   - USB1_UTMIP_XCVR_CFG1_0[UTMIP_XCVR_HS_IREF_CAP] to FUSE_USB_CALIB_0[14:13]
   - USB1_UTMIP_BIAS_CFG0_0[UTMIP_HSSQUELCH_LEVEL] to FUSE_USB_CALIB_0[12:11]
5. Set the following USBOTG register bits to '1' to switch the VBUS detection from hardware operation to software override.
   - USB1_IF_USB_PHY_VBUS_SENSORS_0[B_SESS_VLD_SW_VALUE]
   - USB1_IF_USB_PHY_VBUS_SENSORS_0[B_SESS_VLD_SW_EN]
6. Ensure the following USBOTG register bit is set to '1' during charger detection. This bit should be cleared to '0' after charger detection is completed.
   - USB1_UTMIP_XCVR_CFG0_0[UTMIP_FORCE_PD2_POWERUP]

Step 4

Pad driver switches USB 2.0 related PLLs to under hardware control after pad driver is loaded.

Set up UTMIPLL under Hardware control

1. Enable Hardware Power Sequencer:
   - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLL_ENABLE_POWERUP] = 0
   - CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0[UTMIP_FORCE_PLL_ENABLE_POWERDOWN] = 0
2. If (USB2.0 Controller IP exists in platform), skip to step 4.
3. Program these registers:
   - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_SWCTL] = 0
   - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_PD_INCLUDE] = 1
4. Program these registers:
   - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_CLK_ENABLE_SWCTL] = 0
   - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_USE_LOCKDET] = 1
   - CLK_RST_CONTROLLER_XUSB_PLL_CFG0_0[UTMIPLL_LOCK_DLY] = 0

5. Wait 1 µs
   - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_SEQ_ENABLE] = 1

Set up PLLU under Hardware control

1. Enable Hardware Power Sequencer:
   - CLK_RST_CONTROLLER_PLLU_BASE_0[PLLU_OVERRIDE] = 0
   - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_IDDQ_PD_INCLUDE] = 1
   - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_USE_SWITCH_DETECT] = 1
   - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_CLK_ENABLE_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_CLK_SWITCH_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_USE_LOCKDET] = 1
   - CLK_RST_CONTROLLER_XUSB_PLL_CFG0_0[PLLU_LOCK_DLY] = 0
2. Wait 1 µs
   - CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0[PLLU_SEQ_ENABLE] = 1
     Pad driver enables USB3.0 related PLLs.
3. Enable Hardware Power Sequencer:
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_IDDQ_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SS_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_ENABLE_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SS_SEQ_INCLUDE] = 1
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_USE_LOCKDET] = 1
4. Wait 1 µs.
5. CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SEQ_ENABLE] = 1
6. Enable Hardware Power Sequencer for PLLE:
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_IDDQ_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SS_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_ENABLE_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SS_SEQ_INCLUDE] = 1
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_USE_LOCKDET] = 1
7. Wait 1 µs.
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SEQ_ENABLE] = 1

The pad driver sets up USB3.0 related PLLE.

Set up PLL under Hardware control

PLLREFE

1. The pad driver sets up USB3.0 related PLLREFE in software
   - CLK_RST_CONTROLLER_PLLREFE_MISC_0[PLLREFE_IDDQ]= 0
2. Wait 5 µs
   - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_DIVM]
   - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_DIVN]
   - CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_DIVP]

- CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_KCP]
- CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_KVCO]
- CLK_RST_CONTROLLER_PLLREFE_BASE_0[PLLREFE_ENABLE] = 1

3. Wait for LOCK:
   - CLK_RST_CONTROLLER_PLLREFE_MISC_0[PLLREFE_LOCK]

Note: *There is no Hardware Power Sequencer for PLLREFE.*

PLLEs

The pad driver sets up USB3.0 related PLLE in software.

1. Select XTAL as the source:
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_REF_SEL_PLLREFE] = 0
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_IDDQ_OVERRIDE_VALUE] = 0
2. Wait 5 µs.
3. Program the following registers to generate a low jitter 100 MHz clock:
   - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_MDIV]
   - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_NDIV]
   - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_PLDIV_CML]
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_KCP]
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_KVCO]
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_VREG_CTRL]
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_PTS] = 1
   - CLK_RST_CONTROLLER_PLLE_BASE_0[PLLE_ENABLE] = 1
4. Wait for LOCK:
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_LOCK]
5. Enable SSA:
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_SSCINC]
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_SSCINCINTRV]
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_SSCMAX]
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_SSCINVERT]
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_SSCCENTER]
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_BYPASS_SS] = 0
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_SSCBYP] = 0
6. Wait 300 ns
   - CLK_RST_CONTROLLER_PLLE_SS_CNTL_0[PLLE_INTERP_RESET] = 0
7. Enable Hardware Power Sequencer for PLLE:
   - CLK_RST_CONTROLLER_PLLE_MISC_0[PLLE_IDDQ_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SS_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_ENABLE_SWCTL] = 0
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SS_SEQ_INCLUDE] = 1

- CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_USE_LOCKDET] = 1

8. Wait 1 µs.
   - CLK_RST_CONTROLLER_PLLE_AUX_0[PLLE_SEQ_ENABLE] = 1

UPHY PLLs

Enable PEX_PAD PLL in software:

1. Deassert PLL/Lane resets.
   - CLK_RST_CONTROLLER_RST_DEVICES_Y_0[SWR_PEX_USB_UPHY_RST] = 0
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0[PLL0_CAL_CTRL] = 0x136
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_5_0[PLL0_DCO_CTRL] = 0x2A
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[PLL0_PWR_OVRD] = 1 (Default)
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0[PLL0_CAL_OVRD] = 1
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_OVRD] = 1
2. For the following registers, default values take care of the desired frequency.
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_4_0[REFCLK_SEL]
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_4_0[TXCLKREF_SEL]
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_4_0[TXCLKREF_EN]
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[FREQ_MDIV]
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[FREQ_NDIV]
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[FREQ_PSDIV]
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[PLL0_IDDQ] = 0
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[PLL0_SLEEP] = 0
3. Wait 100 ns.
4. Calibration:
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0[PLL0_CAL_EN] = 1
   - Wait for XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0[PLL0_CAL_DONE] == 1
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0[PLL0_CAL_EN] = 0
   - Wait for XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0[PLL0_CAL_DONE] == 0
5. Enable the PLL (20 µs Lock time)
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[PLL0_ENABLE] = 1
   - Wait for XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[PLL0_LOCKDET_STATUS] == 1
6. RCAL:
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_EN] = 1
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_CLK_EN] = 1
   - Wait for XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_DONE] == 1
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_EN] = 0
   - Wait for XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_DONE] == 0
   - XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_CLK_EN] = 0

Enable SATA_PAD PLL in Software:

1. For Reset deassertion:

- CLK_RST_CONTROLLER_RST_DEVICES_Y_0[SWR_SATA_USB_UPHY_RST] = 0

2. All steps are same as mentioned above for pex pad – Just change the register to: XUSB_PADCTL_UPHY_PLL_S0_CTL_*_0

Note: *FREQ_NDIV and TXCLKREF_SEL should be programmed differently in the above registers for SATA Brick if configured for XUSB.*

Enable Hardware Power Sequencer:

1. For the PEX_pad:
   a. Lift the SWCTLs from the CAR:
      - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_CLK_ENABLE_SWCTL] = 0
      - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_PADPLL_RESET_SWCTL] = 0

   b. Include IDDQ and LOCK-Detect in the hardware sequence.
      - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_PADPLL_SLEEP_IDDQ] = 1
      - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_PADPLL_USE_LOCKDET] = 1

   c. Lift PADCTL overrides
      - XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0[PLL0_PWR_OVRD] = 0
      - XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0[PLL0_CAL_OVRD] = 0
      - XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0[PLL0_RCAL_OVRD] = 0

   d. Wait 1 µs.
      - CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0[XUSBIO_SEQ_ENABLE] = 1

2. For the SATA_pad:
   a. Lift the SWCTLs from CAR
      - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_PADPLL_RESET_SWCTL] = 0

   b. Include IDDQ and LOCK-Detect in the hardware sequence.
      - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_PADPLL_SLEEP_IDDQ] = 1
      - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_PADPLL_USE_LOCKDET] = 1

   c. Lift PADCTL overrides
      - XUSB_PADCTL_UPHY_PLL_S0_CTL_1_0[PLL0_PWR_OVRD] = 0
      - XUSB_PADCTL_UPHY_PLL_S0_CTL_2_0[PLL0_CAL_OVRD] = 0
      - XUSB_PADCTL_UPHY_PLL_S0_CTL_8_0[PLL0_RCAL_OVRD] = 0

   d. Wait 1 µs.
      - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_ENABLE] = 1

Step 5

Pad driver enables platform specific regulators enable power rails to the pads, VBUS, and pull-up voltage to the VBUS control PMIC's EN input.

Pad driver deasserts reset to XUSB pad.

1. Set the following CAR register bit to '0' to deassert reset to PCIe/XUSB pad and SATA/XUSB pad.
   - CLK_RST_CONTROLLER_RST_DEVICES_Y_0[SWR_PEX_USB_UPHY_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_Y_0[SWR_SATA_USB_UPHY_RST]

Pad driver sets up the I/O pins for the VBUS control and over current report.

2. Program the following PINMUX registers to set up the I/O pins for USB 2.0 ports owned by XUSB and USB2, according to the platform specific configuration:
   - PINMUX_AUX_USB_VBUS_EN0_0[E_OD]
   - PINMUX_AUX_USB_VBUS_EN0_0[E_IO_HV]
   - PINMUX_AUX_USB_VBUS_EN0_0[E_INPUT]
   - PINMUX_AUX_USB_VBUS_EN0_0[PARK]
   - PINMUX_AUX_USB_VBUS_EN0_0[TRISTATE]
   - PINMUX_AUX_USB_VBUS_EN0_0[PUPD]
   - PINMUX_AUX_USB_VBUS_EN1_0[E_OD]
   - PINMUX_AUX_USB_VBUS_EN1_0[E_IO_HV]
   - PINMUX_AUX_USB_VBUS_EN1_0[E_INPUT]
   - PINMUX_AUX_USB_VBUS_EN1_0[PARK]
   - PINMUX_AUX_USB_VBUS_EN1_0[TRISTATE]
   - PINMUX_AUX_USB_VBUS_EN1_0[PUPD]

   Pad driver sets up the VGPIO for VBUS and ID status reporting used by the device and OTG modes of XUSB and USB2.
3. Program the following XUSB PADCTL registers to use local override for VBUS and ID status reporting
   - XUSB_PADCTL_USB2_VBUS_ID_0[ID_SOURCE_SELECT] to ‘ID_OVERRIDE’
   - XUSB_PADCTL_USB2_VBUS_ID_0[VBUS_SOURCE_SELECT] to ‘VBUS_OVERRIDE’
4. Write ‘1’ to the following XUSB PADCTL registers to clear false reporting of VBUS and ID status changes
   - XUSB_PADCTL_USB2_VBUS_ID_0[IDDIG_ST_CHNG]
   - XUSB_PADCTL_USB2_VBUS_ID_0[VBUS_VALID_ST_CHNG]

   Pad driver assigns the USB ports to the controllers, then programs the port capabilities and pad parameters of ports assigned to XUSB after booted to OS.
5. Set the following CAR register bits to ‘1’ to assert reset to USB2
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0[SWR_USBD_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0[SWR_USB2_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0[SWR_USB3_RST]
6. Program the following XUSB PADCTL registers to assign the USB2.0 ports to XUSB or USB2, according to the platform specific configuration. (Note: Only OTG and HSIC Pad Port 0 could be assigned to USB2 controller.)
   - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_HSIC_PAD_PORT0]
   - XUSB_PADCTL_USB2_PAD_MUX_0[HSIC_PAD_TRK]
   - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_ULPI_PAD_PORT]
   - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT2]
   - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT1]
   - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT0]
   - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_BIAS_PAD]
7. Program the following XUSB PADCTL registers to assign the port capabilities for USB2.0 ports owned by XUSB, according to the platform specific configuration:
   - XUSB_PADCTL_USB2_PORT_CAP_0[PORT3_CAP] for host/disabled

- XUSB_PADCTL_USB2_PORT_CAP_0[PORT3_INTERNAL] for whether its internal port
- XUSB_PADCTL_USB2_PORT_CAP_0[PORT2_CAP] for host/disabled
- XUSB_PADCTL_USB2_PORT_CAP_0[PORT2_INTERNAL] for whether its internal port
- XUSB_PADCTL_USB2_PORT_CAP_0[PORT1_CAP] for host/disabled
- XUSB_PADCTL_USB2_PORT_CAP_0[PORT1_INTERNAL] for whether its internal port
- XUSB_PADCTL_USB2_PORT_CAP_0[PORT0_CAP] for host/device/OTG/disabled
- XUSB_PADCTL_USB2_PORT_CAP_0[PORT0_INTERNAL] for whether its internal port

8. Program the following XUSB PADCTL registers to 'OC_DETECTION_DISABLED' to disable the over current signal mapping for USB 2.0 ports owned by XUSB and USB2:
   - XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER1_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT3_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT2_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT1_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT0_OC_PIN]
   - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE3_OC_MAP]
   - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE2_OC_MAP]
   - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE1_OC_MAP]
   - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE0_OC_MAP]
9. Program the following XUSB PADCTL registers to assign the SuperSpeed port mapping to USB2.0 ports owned by XUSB, where the SuperSpeed ports inherit their port capabilities from the USB2.0 ports they mapped to, according to the platform specific configuration:
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT3_MAP] to USB2.0 Port 3/2/1/0 or Disabled
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT3_INTERNAL] for whether its internal port
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT2_MAP] to USB2.0 Port 3/2/1/0 or Disabled
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT2_INTERNAL] for whether its internal port
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT1_MAP] to USB2.0 Port 3/2/1/0 or Disabled
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT1_INTERNAL] for whether its internal port
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT0_MAP] to USB2.0 Port 3/2/1/0 or Disabled
   - XUSB_PADCTL_SS_PORT_MAP_0[PORT0_INTERNAL] for whether its internal port
10. Program the following XUSB PADCTL registers to keep the UPHY in IDDQ and under sleep state before changing the LANE assignments, where *$LN* = P0, P1, P2, P3, P4, P5, P6, S0:
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_IDDQ_OVRD_RANGE] to '1'
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_IDDQ_RANGE] to '1'
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_SLEEP_RANGE] to '3'
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_PWR_OVRD_RANGE] to '1'
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_IDDQ_OVRD_RANGE] to '1'
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_IDDQ_RANGE] to '1'
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_SLEEP_RANGE] to '3'
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_PWR_OVRD_RANGE] to '1'
11. Program the following XUSB PADCTL registers to assign the UPHY lanes to XUSB, PCIe, or SATA, according to the platform specific configuration

- XUSB_PADCTL_USB3_PAD_MUX_0[SATA_PAD_LANE0]
- XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE6]
- XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE5]
- XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE4]
- XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE3]
- XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE2]
- XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE1]
- XUSB_PADCTL_USB3_PAD_MUX_0[PCIE_PAD_LANE0]

12. Program the following XUSB PADCTL registers to bring the UPHY out of IDDQ after changing the LANE assignments, where *$LN* = P0, P1, P2, P3, P4, P5, P6, S0:
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_IDDQ_OVRD_RANGE] to ‘0’
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_IDDQ_RANGE] to ‘1’
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_SLEEP_RANGE] to ‘3’
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[TX_PWR_OVRD_RANGE] to ‘0’
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_IDDQ_OVRD_RANGE] to ‘0’
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_IDDQ_RANGE] to ‘1’
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_SLEEP_RANGE] to ‘3’
    - XUSB_PADCTL_UPHY_MISC_PAD_*$LN*_CTL_2_0[RX_PWR_OVRD_RANGE] to ‘0’
13. Program the following XUSB PADCTL registers to assign the static UPHY pad and PLL parameters according to the platform specific configuration, where *$LN* = P0, P1, P2, P3, P4, P5, P6, S0:
    - XUSB_PADCTL_UPHY_PLL_P0_CTL1_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL2_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL3_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL4_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL5_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL6_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL7_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL8_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL9_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL10_0
    - XUSB_PADCTL_UPHY_PLL_P0_CTL11_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL1_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL2_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL3_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL4_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL5_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL6_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL7_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL8_0
    - XUSB_PADCTL_UPHY_PLL_S0_CTL9_0

- XUSB_PADCTL_UPHY_PLL_S0_CTL10_0
- XUSB_PADCTL_UPHY_PLL_S0_CTL11_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_1_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_2_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_3_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_4_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_5_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_6_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_7_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_8_0
- XUSB_PADCTL_UPHY_MISC_PAD_$LN_CTL_9_0

14. Program the following XUSB PADCTL registers to assign the static UPHY pad parameters of ports owned by XUSB, according to the platform specific configuration:
    - XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_1_0
    - XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_2_0
    - XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_3_0
    - XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_4_0
    - XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_5_0
    - XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_6_0
    - XUSB_PADCTL_UPHY_USB3_PAD0_CTL_0_0
    - XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_1_0
    - XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_2_0
    - XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_3_0
    - XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_4_0
    - XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_5_0
    - XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_6_0
    - XUSB_PADCTL_UPHY_USB3_PAD1_CTL_0_0
    - XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_1_0
    - XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_2_0
    - XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_3_0
    - XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_4_0
    - XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_5_0
    - XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_6_0
    - XUSB_PADCTL_UPHY_USB3_PAD2_CTL_0_0
    - XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_1_0
    - XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_2_0
    - XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_3_0
    - XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_4_0
    - XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_5_0

- XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_6_0
- XUSB_PADCTL_UPHY_USB3_PAD3_CTL_0_0

15. Program the following XUSB PADCTL registers to assign the static USB2.0 pad parameters of ports owned by XUSB, according to the platform specific configuration:
    - XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0
    - XUSB_PADCTL_USB2_OTG_PAD0_CTL_1_0
    - XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0
    - XUSB_PADCTL_USB2_OTG_PAD1_CTL_1_0
    - XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0
    - XUSB_PADCTL_USB2_OTG_PAD2_CTL_1_0
    - XUSB_PADCTL_USB2_OTG_PAD3_CTL_0_0
    - XUSB_PADCTL_USB2_OTG_PAD3_CTL_1_0
    - XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0
    - XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0
    - XUSB_PADCTL_HSIC_PAD0_CTL_0_0
    - XUSB_PADCTL_HSIC_PAD0_CTL_1_0
    - XUSB_PADCTL_HSIC_PAD0_CTL_2_0
    - XUSB_PADCTL_HSIC_PAD_TRK_CTL_0_0
    - XUSB_PADCTL_HSIC_STRB_TRIM_CONTROL_0
16. Program the following XUSB PADCTL register bits to ‘0’ to disable power down of the USB2.0 of ports owned by XUSB, according to the platform specific configuration.
    - XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0[PD]
    - XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0[PD]
    - XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0[PD]
    - XUSB_PADCTL_USB2_OTG_PAD3_CTL_0_0[PD]
    - XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0[PD2]
    - XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0[PD2]
    - XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0[PD2]
    - XUSB_PADCTL_USB2_OTG_PAD3_CTL_0_0[PD2]
    - XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0[PD_ZI]
    - XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0[PD_ZI]
    - XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0[PD_ZI]
    - XUSB_PADCTL_USB2_OTG_PAD3_CTL_0_0[PD_ZI]
    - XUSB_PADCTL_USB2_OTG_PAD0_CTL_1_0[PD_DR]
    - XUSB_PADCTL_USB2_OTG_PAD1_CTL_1_0[PD_DR]
    - XUSB_PADCTL_USB2_OTG_PAD2_CTL_1_0[PD_DR]
    - XUSB_PADCTL_USB2_OTG_PAD3_CTL_1_0[PD_DR]
    - XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0[PD]
    - XUSB_PADCTL_HSIC_PAD0_CTL_0_0[PD_TX_DATA0]
    - XUSB_PADCTL_HSIC_PAD0_CTL_0_0[PD_TX_STROBE]

- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[PD_RX_DATA0]
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[PD_RX_STROBE]
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[PD_ZI_DATA0]
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[PD_ZI_STROBE]

17. Program the following XUSB PADCTL register bits to '0' to release the XUSB SS wake logic state latching
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN_EARLY]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_VCORE_DOWN]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN_EARLY]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_VCORE_DOWN]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN_EARLY]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_VCORE_DOWN]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN_EARLY]
    - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_VCORE_DOWN]
18. Perform USB2.0 and HSIC pad tracking. (Refer to Section 22.8.10: USB2 Pad Tracking Programming.)

Note: *To avoid confliction or ambiguity, the XUSB and USB2 controllers should be under reset when updating the port assignments.*

Step 6

Pad driver programs the clocks and deasserts the resets to the controllers

1. Set the following CAR register bits to '1' to enable the clocks to XUSB
    - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB]
2. Set the following CAR register bits to '1' to enable the clocks to individual XUSB partitions
    - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_HOST]
    - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_DEV]
    - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS]
3. Program the following CAR register bits to set the source of XUSB clocks, where PLLP_OUT0 runs at 408 MHz.
    - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_HOST_0[XUSB_CORE_HOST_CLK_SRC] to 'PLLP_OUT0'
    - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_HOST_0[XUSB_CORE_HOST_CLK_DIVISOR] to '0x6'
    - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_DEV_0[XUSB_CORE_DEV_CLK_SRC] to 'PLLP_OUT0'
    - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_DEV_0[XUSB_CORE_DEV_CLK_DIVISOR] to '0x6'
    - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FALCON_0[XUSB_FALCON_CLK_SRC] to 'PLLP_OUT0'
    - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FALCON_0[XUSB_FALCON_CLK_DIVISOR] to '0x2'
    - CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FS_0[XUSB_FS_CLK_SRC] to 'FO_48M'

- CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FS_0[XUSB_FS_CLK_DIVISOR] to ‘0x0’
- CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_SS_0[XUSB_SS_CLK_SRC] to ‘HSIC_480’
- CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_SS_0[XUSB_SS_CLK_DIVISOR] to ‘0x6’

4. Set the following CAR register bits to ‘0’ to deassert reset to XUSB
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST]
5. Set the following CAR register bits to ‘0’ to deassert reset to USB2
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0[SWR_USBD_RST]
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0[SWR_USB2_RST]

Step 7

Pad driver programs the capability and pad parameters of ports assigned to USB2 controllers.

1. Program the following USB2 registers of each controller to assign the port capabilities of ports owned by USB2, according to the platform specific configuration:
   - USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0[PTS] for HSIC or UTMI
   - USB2_CONTROLLER_USB2D_USBMODE_0[CM] for Host Mode or Device Mode
2. Program the following USB2 registers of each controller to assign the pad parameters of ports owned by USB2 according to the platform specific configuration:
   - UHSIC_PLL_CFG0_0
   - UHSIC_HSRX_CFG0_0
   - UHSIC_HSRX_CFG1_0
   - UHSIC_TX_CFG0_0
   - UHSIC_MISC_CFG0_0
   - UHSIC_MISC_CFG1_0
   - UHSIC_PADS_CFG0_0
   - UHSIC_PADS_CFG1_0
   - UHSIC_CMD_CFG0_0
   - UHSIC_STAT_CFG0_0
   - UHSIC_SPARE_CFG0_0
   - UTMIP_PLL_CFG0_0
   - UTMIP_PLL_CFG1_0
   - UTMIP_XCVR_CFG0_0
   - UTMIP_XCVR_CFG1_0
   - UTMIP_BIAS_CFG0_0
   - UTMIP_BIAS_CFG1_0
   - UTMIP_HSRX_CFG0_0
   - UTMIP_HSRX_CFG1_0
   - UTMIP_FSLSRX_CFG0_0
   - UTMIP_FSLSRX_CFG1_0
   - UTMIP_TX_CFG0_0

- UTMIP_MISC_CFG0_0
- UTMIP_MISC_CFG1_0
- UTMIP_SPARE_CFG0_0

Notes:

- *In Tegra X1, USB2 Controller #1 always uses UTMI, USB2 Controller #2 always uses HSIC.*
- *Only USB2 Controller #1 may be set to Device Mode.*

Step 8

Pad driver brings the UPHY out of IDDQ.

1. Program the following XUSB PADCTL registers to '1' to bring specific lanes of UPHY out of IDDQ. Only lanes that are used in the platform are required to be bring out of IDDQ:
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK6]
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK5]
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4]
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3]
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2]
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1]
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0]
2. Alternatively, program the following XUSB PADCTL register to '0' to bring all lanes of IOPHY out of IDDQ:
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE]

Pad driver releases the always on pad muxing logic state latching.

3. Program the following XUSB PADCTL register bits
4. Wait 1 µs
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[AUX_MUX_LP0_CLAMP_EN] to '0'
5. Wait 100 µs
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[AUX_MUX_LP0_CLAMP_EN_EARLY] to '0'
6. Wait 100 µs
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[AUX_MUX_LP0_VCORE_DOWN] to '0'

Pad driver enables the VBUS to the USB ports.

7. Program the following XUSB PADCTL registers to assign the over current signal mapping for USB 2.0 ports owned by XUSB and USB2, according to the platform specific configuration:
   - XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER1_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT3_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT2_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT1_OC_PIN]
   - XUSB_PADCTL_USB2_OC_MAP_0[PORT0_OC_PIN]
   - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE3_OC_MAP]
   - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE2_OC_MAP]

- XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE1_OC_MAP]
- XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE0_OC_MAP]

8. Write ‘1’ to the following XUSB PADCTL register bits to clear possible false reporting of over current events before the over current signal mappings are properly programmed:
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED3]
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED2]
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED1]
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED0]
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED_VBUS_PAD3]
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED_VBUS_PAD2]
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED_VBUS_PAD1]
   - XUSB_PADCTL_OC_DET_0[OC_DETECTED_VBUS_PAD0]
9. Wait 1 µs
10. Set the following XUSB PADCTL register bits to ‘1’ to enable the VBUS of the host ports
    - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE3]
    - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE2]
    - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE1]
    - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE0]

Step 9

xHCI PEP driver initializes IPFS registers.

1. Program the following XUSB Host IPFS registers to allow software accesses to XUSB Host’s MMIO registers:
   - XUSB_HOST_AXI_BAR0_START_0[AXI_BAR0_START] to ‘0x70090’
   - XUSB_HOST_AXI_BAR0_SZ_0[AXI_BAR0_SIZE] to ‘0x00008’
   - XUSB_HOST_FPCI_BAR0_0[FPCI_BAR0_START] to ‘0x0010000’
   - XUSB_HOST_FPCI_BAR0_0[FPCI_BAR0_ACCESS_TYPE] to ‘0’
2. Program the following XUSB Host IPFS register to enable the XUSB host:
   - XUSB_HOST_CONFIGURATION_0[EN_FPCI] to ‘1’

xHCI Device driver initializes IPFS registers.

3. Program the following XUSB Device IPFS registers to allow software accesses to XUSB Device’s MMIO registers:
   - XUSB_DEV_AXI_BAR0_START_0[AXI_BAR0_START] to ‘0x700d0’
   - XUSB_DEV_AXI_BAR0_SZ_0[AXI_BAR0_SIZE] to ‘0x00008’
   - XUSB_DEV_FPCI_BAR0_0[FPCI_BAR0_START] to ‘0x0010000’
   - XUSB_DEV_FPCI_BAR0_0[FPCI_BAR0_ACCESS_TYPE] to ‘0’
4. Program the following XUSB Host IPFS register to enable the XUSB host:
   - XUSB_HOST_CONFIGURATION_0[EN_FPCI] to ‘1’

Note: *XUSB Host’s and Device’s configuration address mappings are hardcoded in IPFS starting from offset 0x8000.*

---

xHCI PEP driver initializes XUSB Host’s configuration registers.

5. Program the following XUSB configuration registers to initialize XUSB:
   - NV_PROJ__XUSB_CFG_1_BUS_MASTER to ‘1’
   - NV_PROJ__XUSB_CFG_1_MEMORY_SPACE to ‘1’
   - NV_PROJ__XUSB_CFG_4_BASE_ADDRESS to ‘0x02000’

xHCI Device driver initializes XUSB Host’s configuration registers.

6. Program the following XUSB configuration registers to initialize XUSB:
   - NV_PROJ__XUSB_DEV_CFG_1_BUS_MASTER to ‘1’
   - NV_PROJ__XUSB_DEV_CFG_1_MEMORY_SPACE to ‘1’
   - NV_PROJ__XUSB_DEV_CFG_4_BASE_ADDRESS to ‘0x02000’

Step 10

xHCI PEP driver loads XUSB Host firmware.

During system cold boot, the Boot Loader is responsible for triggering the hardware to load the firmware into Falcon. During exit from power gating either when the XUSB was power gated or when the system exits from LP, the xHCI driver for Linux OS and xHCI driver power management PEP for Windows will be responsible to trigger the hardware to load the firmware into Falcon.

The sequence of steps performed by the FW boot loader is as follows:

1. Copy the DFI image into system memory (physically contiguous). DFI contains the configuration table, FW code, and Falcon bootstrap code.
2. Cold Boot:
   a. The Boot Loader sets the CONTEXT_SAVED bit to “0”.

   Note: *This bit is reset to “0” so the Boot Loader can ignore this step.*

3. Exit from Power Gating:
   - Linux xHCI driver or Windows xHCI PEP sets the CONTEXT_SAVED bit to “0” or “1” according to the state it read from the CONTEXT_SAVED bit before power gating the XUSB.
   - Linux xHCI driver or Windows xHCI PEP sets the PORT_STATE_SAVED and FS_PORT_SPEED_SAVED fields according to the state it read from these same fields before power gating XUSB.
4. Program the system memory address at which the FW code starts to NV_PROJ__XUSB_CSB_MEMPOOL_ILOAD_BASE_LO, NV_PROJ__XUSB_CSB_MEMPOOL_ILOAD_BASE_HI, and NV_PROJ__XUSB_CSB_MEMPOOL_ILOAD_ATTR_SIZE.

   Notes:
   - *FW code address program in the above step is = Start address of DFI + 256.*
   - *The value to be programmed to ILOAD_ATTR_SIZE is stored in the DFIOFFSET field present with in the configuration table of DFI image.*

5. Invalidate all entries in the L2IMEM by writing ‘L2IMEM_INVALIDATE_ALL’ to ‘ACTION’ field in the NV_PROJ__XUSB_CSB_MEMPOOL_L2IMEMOP_TRIG register.
6. Program the L2IMEM register as follows to fetch the complete bootstrap into L2IMEM:
   - Program the 256 byte aligned offset of the FW bootstrap code (in DFI image) in the system memory to NV_PROJ__XUSB_CSB_MEMPOOL_L2IMEMOP_SIZE_OFFSET. The offset corresponds to bootstrap location from the start of the complete FW code.
   - Program the number of code pages (multiples of 256B) to copy from system memory from start of the FW bootstrap code to NV_PROJ__XUSB_CSB_MEMPOOL_L2IMEMOP_SIZE_SRC_COUNT

- Program the L2IMEM action to be performed as ‘L2IMEM_LOAD_LOCKED_RESULT’ into NV_PROJ__XUSB_CSB_MEMPOOL_L2IMEMOP_TRIG_ACTION (DEST_INDEX will be zero).

7. Reserve the required number of IMEM blocks by writing to ‘IMFILLCTL’ register.
8. Enable the auto-fill mode for the bootstrap code range by programming ‘IMFILLRNG1’ register. The low tag will be ‘Boot Tag’ present in the configuration table and high tag will be ‘Boot tag + FW bootstrap code size).
9. Program the BOOTVEC register with the location of IMEM at which Falcon boot code is copied. The value programmed will be the “Boot tag” value present in the DFI configuration table.
10. Start the Falcon by writing to the STARTCPU field in the CPUCTL register.

22.8.5 LP0

22.8.5.1 LP0 Entry

Step 1

The EHCI driver performs context save operations.

The xHCI driver performs a context save operation as described in Section 4.23.2 of the xHCI specification.

The xHCI PEP driver performs an XUSB specific context save operation.

The xHCI PEP driver performs an XUSB IPFS specific context save operation.

- Read and store the value of the following registers
  - XUSB_{HOST,DEV}_MSI_BAR_SZ_0
  - XUSB_{HOST,DEV}_MSI_AXI_BAR_ST_0
  - XUSB_{HOST,DEV}_MSI_FPCI_BAR_ST_0
  - XUSB_{HOST,DEV}_MSI_VEC0_0
  - XUSB_{HOST,DEV}_MSI_EN_VEC0_0
  - XUSB_{HOST,DEV}_FPCI_ERROR_MASKS_0
  - XUSB_{HOST,DEV}_INTR_MASK_0
  - XUSB_{HOST,DEV}_IPFS_INTR_ENABLE_0
  - XUSB_{HOST,DEV}_UFPCI_CONFIG_0
  - XUSB_{HOST,DEV}_CLKGATE_HYSTERESIS_0
  - XUSB_{HOST,DEV}_XUSB_HOST_MCCIF_FIFOCTRL_0

Step 2

The System Power Management driver programs the PMC USB2.0 sleepwalk logic as described in the “PMC Programming” section.

The System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 and XUSB registers ports accordingly.

The System Power Management driver enables wake events from USB ports.

- Set the following PMC register bits to ‘1’ to set the wake signal active level to ‘HIGH’
  - APBDEV_PMC_WAKE2_LVL_0[7] for USB2.0 port 0 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[8] for USB2.0 port 1 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[9] for USB2.0 port 2 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[10] for USB2.0 port 3 wakeup
  - APBDEV_PMC_WAKE2_LVL_0[11] for HSIC port 0 wakeup

- APBDEV_PMC_WAKE2_LVL_0[12] for USB3.0 ports wakeup

- Set the following PMC register bits to '1' to enable USB wake events
  - APBDEV_PMC_WAKE2_MASK_0[7] for USB2.0 port 0 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[8] for USB2.0 port 1 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[9] for USB2.0 port 2 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[10] for USB2.0 port 3 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[11] for HSIC port 10 wakeup
  - APBDEV_PMC_WAKE2_MASK_0[12] for USB3.0 ports wakeup

Step 3

The System Power Management driver setup the VBUS_ENABLE I/O pins as GPIO with wake events enabled to enable wake on overcurrent.

- Set the following XUSB PADCTL registers to 'OC_DETECTION_DISABLED' to disable over current wake events reporting from XUSB PADCTL
  - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE3_OC_MAP]
  - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE2_OC_MAP]
  - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE1_OC_MAP]
  - XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE0_OC_MAP]
- Program the following GPIO registers to enable the I/O pins to generate events when over current events are reported with signals asserting to logical low
  - GPIO_CNF_1_3[BIT_5] to 'GPIO'
  - GPIO_CNF_1_3[BIT_4] to 'GPIO'
  - GPIO_OE_1_3[BIT_5] to 'TRI_STATE'
  - GPIO_OE_1_3[BIT_4] to 'TRI_STATE'
  - GPIO_INT_ENB_1_3[BIT_5] to 'ENABLE'
  - GPIO_INT_ENB_1_3[BIT_4] to 'ENABLE'
  - GPIO_INT_LVL_1_3[EDGE_5] to 'EDGE-Triggered'
  - GPIO_INT_LVL_1_3[EDGE_4] to 'EDGE-Triggered'
  - GPIO_INT_LVL_1_3[BIT_5] to 'LOW'
  - GPIO_INT_LVL_1_3[BIT_4] to 'LOW'
- Set the following PMC register bits to '1' to set the wake signal active level to 'HIGH' and enable the wake events
  - APBDEV_PMC_WAKE2_LVL_0[23] for VBUS_ENABLE1
  - APBDEV_PMC_WAKE2_LVL_0[22] for VBUS_ENABLE0
  - APBDEV_PMC_WAKE2_MASK_0[23] for VBUS_ENABLE1
  - APBDEV_PMC_WAKE2_MASK_0[22] for VBUS_ENABLE0

Step 4

The pad driver asserts reset to XUSB pad.

- Set the following CAR register bits to '1' to assert reset to PCIe/XUSB pad and SATA/XUSB pad.
  - CLK_RST_CONTROLLER_RST_DEVICES_Y_0[SWR_PEX_USB_UPHY_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_Y_0[SWR_SATA_USB_UPHY_RST]

The pad driver disables DVDD power of the XUSB pad by turning off regulators with platform specific means.

Step 5

The system software puts the system in LP0.

22.8.5.2 LP0 Exit

After exiting LP0, the PMC will restore the partition power gating to the state before LP0 entry. This means if a partition was power gated before entering LP0, that partition would stay power gated after exiting LP0, and only partitions that were powered before entering LP0 would have their power restored after exiting LP0.

Step 1

The Boot ROM enables USB related PLLs with software override.

Refer to Step 1 of “Cold Boot” for the programming sequence.

Step 2

The Boot ROM deasserts reset to XUSB PADCTL block.

Refer to Step 2 of “Cold Boot” for the programming sequence.

Step 3

The Boot ROM performs battery charging operations through USB2 Controller.

Refer to Step 3 of “Cold Boot” for the programming sequence.

Step 4

The LP0 exit sequence is described in Chapter 12: Power Management Controller of this TRM.

Step 5

The pad driver switches USB related PLLs to under hardware control.

Refer to Step 7 of “Cold Boot” for the programming sequence.

Step 6

The pad driver checks the VBUS_ENABLE I/O pins for over current events and disables wake event reporting.

- Read the following PMC register bits
  - APBDEV_PMC_WAKE2_STATUS_0[23] for VBUS_ENABLE1
  - APBDEV_PMC_WAKE2_STATUS_0[22] for VBUS_ENABLE0
- Program the following GPIO registers to enable the I/O pins to generate events when overcurrent events are reported with signals asserting to logical low:
  - GPIO_CNF_1_3[BIT_5] to ‘SPIO’
  - GPIO_CNF_1_3[BIT_4] to ‘SPIO’
  - GPIO_INT_ENB_1_3[BIT_5] to ‘DISABLE’
  - GPIO_INT_ENB_1_3[BIT_4] to ‘DISABLE’

The pad driver forwards the overcurrent events to XUSB or USB2 if they occurred during LP0.

- Program the following XUSB PADCTL registers to enable software override of over current wake events
  - XUSB_PADCTL_OC_DET_0[SET_OC_DETECTED0]
- Program the following XUSB PADCTL registers to ‘OC_DETECTED0’ to report the overcurrent wake events to XUSB or USB2 ports, according to the platform specific configuration:

- XUSB_PADCTL_SNPS_OC_MAP_0[CONTROLLER1_OC_PIN]
- XUSB_PADCTL_USB2_OC_MAP_0[PORT3_OC_PIN]
- XUSB_PADCTL_USB2_OC_MAP_0[PORT2_OC_PIN]
- XUSB_PADCTL_USB2_OC_MAP_0[PORT1_OC_PIN]
- XUSB_PADCTL_USB2_OC_MAP_0[PORT0_OC_PIN]
- XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE3_OC_MAP]
- XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE2_OC_MAP]
- XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE1_OC_MAP]
- XUSB_PADCTL_VBUS_OC_MAP_0[VBUS_ENABLE0_OC_MAP]

Note: *Once the overcurrent condition has been removed, pad driver should set the OC MAP registers to their original mappings and clear the SET OC DETECTED0 bit.*

Step 7

The pad driver assigns the USB port to the controllers, then programs the port capabilities and pad parameters of ports assigned to XUSB according to the platform specific configuration.

Refer to Step 8, #4 through #7, of “Cold Boot” for the programming sequence.

Step 8

The pad driver programs the clocks and deasserts the resets to the controllers.

1. Refer to Step 9 of “Cold Boot” for the start of the programming sequence.
2. Wait 1 µs.
3. Program the following XUSB PADCTL register bits to ‘0’ to release the XUSB SS wake logic state latching
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_VCORE_DOWN]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_VCORE_DOWN]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_VCORE_DOWN]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_VCORE_DOWN]
4. Wait 100 µs.
5. Program the following XUSB PADCTL register bits to ‘0’ to release the XUSB SS wake logic state latching
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN_EARLY]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN_EARLY]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN_EARLY]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN_EARLY]
6. Wait 100 µs.
7. Program the following XUSB PADCTL register bits to ‘0’ to release the XUSB SS wake logic state latching
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN]
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN]

Note: *If XUSB was in ELPG before entering LP0 and the LP0 exit is not due to wake event from ports belong to XUSB, the pad driver should keep XUSB in the ELPG state and only bring XUSB out of ELPG when wake events are detected for the ports owned by XUSB.*

Step 9

The pad driver programs the capability and pad parameters of ports owned by USB2 controllers.

Refer to Step 10 of “Cold Boot” for the USB2 port capability and pad/PLL parameter programming sequence.

Step 10

The pad driver enables the VBUS to the USB ports if VBUS was disabled during LP0.

Refer to Step 11 of “Cold Boot” for the programming sequence.

Step 11

The xHCI PEP driver performs the XUSB specific context restore operation, if XUSB was not in ELPG before entering LP0 or the LP0 exit is due to wake event from ports of XUSB.

The xHCI PEP driver performs XUSB IPFS and XUSB register initialization as described below:

xHCI PEP driver initializes IPFS registers

1. Program the following XUSB Host IPFS registers to allow software accesses to XUSB Host’s MMIO registers:
   - XUSB_HOST_AXI_BAR0_START_0[AXI_BAR0_START] to ‘0x70090’
   - XUSB_HOST_AXI_BAR0_SZ_0[AXI_BAR0_SIZE] to ‘0x00008’
   - XUSB_HOST_FPCI_BAR0_0[FPCI_BAR0_START] to ‘0x0010000’
   - XUSB_HOST_FPCI_BAR0_0[FPCI_BAR0_ACCESS_TYPE] to ‘0’
2. Program the following XUSB Host IPFS register to enable the XUSB host:
   - XUSB_HOST_CONFIGURATION_0[EN_FPCI] to ‘1’

xHCI Device driver initializes IPFS registers

3. Program the following XUSB Device IPFS registers to allow software accesses to XUSB Device’s MMIO registers:
   - XUSB_DEV_AXI_BAR0_START_0[AXI_BAR0_START] to ‘0x700d0’
   - XUSB_DEV_AXI_BAR0_SZ_0[AXI_BAR0_SIZE] to ‘0x00008’
   - XUSB_DEV_FPCI_BAR0_0[FPCI_BAR0_START] to ‘0x0010000’
   - XUSB_DEV_FPCI_BAR0_0[FPCI_BAR0_ACCESS_TYPE] to ‘0’
4. Program the following XUSB Host IPFS register to enable the XUSB host:
   - XUSB_HOST_CONFIGURATION_0[EN_FPCI] to ‘1’

Note: *XUSB Host’s and Device’s configuration address mappings are hardcoded in IPFS starting from offset 0x8000.*

xHCI PEP driver initializes XUSB Host’s configuration registers

5. Program the following XUSB configuration registers to initialize XUSB:
   - NV_PROJ__XUSB_CFG_1_BUS_MASTER to ‘1’
   - NV_PROJ__XUSB_CFG_1_MEMORY_SPACE to ‘1’
   - NV_PROJ__XUSB_CFG_4_BASE_ADDRESS to ‘0x02000’

xHCI Device driver initializes XUSB Host's configuration registers

6. Program the following XUSB configuration registers to initialize XUSB:
   - NV_PROJ__XUSB_DEV_CFG_1_BUS_MASTER to '1'
   - NV_PROJ__XUSB_DEV_CFG_1_MEMORY_SPACE to '1'
   - NV_PROJ__XUSB_DEV_CFG_4_BASE_ADDRESS to '0x02000'

Step 12

The xHCI PEP Driver loads XUSB firmware. Refer to Step 12 in "Cold Boot", if XUSB was not in ELPG before entering LP0 and the LP0 exit is not due to wake event from ports of XUSB.

Step 13

The EHCI driver performs context restore operation.

The xHCI driver performs a context restore operation as described in Section 4.23.2 of the xHCI specification, If XUSB was not in ELPG before entering LP0 and the LP0 exit is not due to wake event from ports of XUSB.

Step 14

The System Power Management driver disables the PMC USB2.0 sleepwalk logic.

The xHCI PEP driver disables the wake event detections.

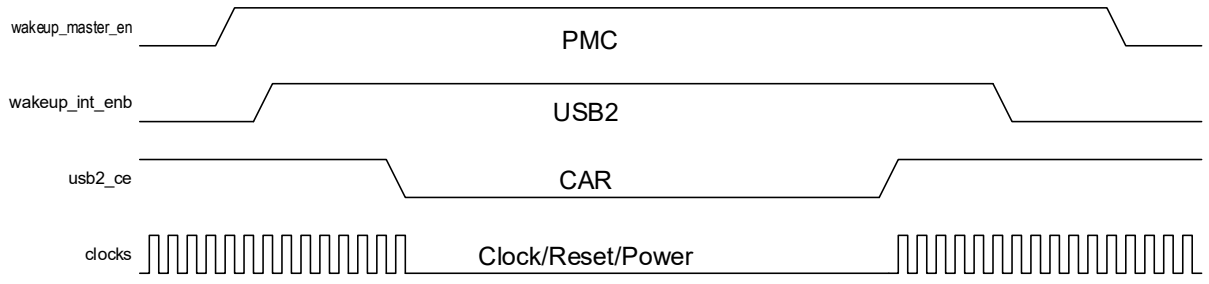
1. Set the following registers to '0' to disable the wake detection:
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
   - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P0]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P1]
   - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P1]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
   - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P2]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_MASTER_ENABLE_P3]
   - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P3]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
   - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UHSIC_LINE_WAKEUP_EN_P0]
2. Set the following registers to '0' to switch the electric control of the USB2.0 pad to XUSB or USB2:
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P0]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P0]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P0]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P1]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P1]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P1]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P2]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P2]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P2]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_FSLS_USE_PMC_P3]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_PCTRL_USE_PMC_P3]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_TCTRL_USE_PMC_P3]

3. Program the following registers to ‘NONE’ to disable wake event triggers of sleepwalk logic:
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P0]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P1]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P2]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_WAKE_VAL_P3]
   - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_WAKE_VAL_P0]
4. Set the following registers to ‘1’ to power down the line state detectors of the pad:
   - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P0]
   - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P0]
   - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P1]
   - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P1]
   - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P2]
   - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P2]
   - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P3]
   - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P3]
   - APBDEV_PMC_USB_AO_0[DATA0_VAL_PD_P0]
   - APBDEV_PMC_USB_AO_0[DATA1_VAL_PD_P0]
   - APBDEV_PMC_USB_AO_0[STROBE_VAL_PD_P0]
5. Write ‘1’ to the following registers to clear alarm of the sleepwalk logic:
   - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P0]
   - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P1]
   - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P2]
   - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P3]
   - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UHSIC_CLR_WAKE_ALARM_P0]

22.8.6 USB2 Controller Clock Gating

USB2 controllers supports controller clock gating, where the port wakeup events are detected via the PMC sleepwalk logic. The following figure illustrates the signal sequences to enable USB2 controller clock gating wake event detection before its clocks are gated.

Figure 57: Signals to Enable USB2 Controller Clock Gating Wake Event Detection



22.8.6.1 Enable Controller Clock Gating

Step 1

The System Power Management driver programs the PMC USB2.0 sleepwalk logic as described in Section 22.8.9.2: Enable PMC Sleepwalk Logic.

The System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

> ***Note:*** *The line wakeup event from PMC is shared between XUSB and USB2 controllers. There is no per controller masking for the line wakeup event, and when the sleepwalk logic triggers the event, both XUSB and USB2 will log the status even when they are not power gated or clock gated. Thus the system power management driver should ensure the wake status is cleared before enabling the sleepwalk logic.*

Step 2

The System Power Management driver can be used to enable the USB2 controller wakeup interrupt.

- Set the following USB2 register bits to '1' to enable the interrupt.
  - USB2_UTMIP_PMC_WAKEUP0_0[UTMIP_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #1
  - UHSIC_PMC_WAKEUP0_0[UHSIC_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #2

Step 3

The System Power Management driver disables the USB2 controller clocks.

- Set the following CAR register bits to '1' to disable the clocks.
  - CLK_RST_CONTROLLER_CLK_ENB_L_CLR_0[CLR_CLK_ENB_USBD] for USB2 #1
  - CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0[CLR_CLK_ENB_USB2] for USB2 #2

Step 4

The System Power Management driver puts UTMIPLL to IDDQ if all non-HSIC USB2.0 ports are assigned to USB2 controllers and all of these controllers are in reset or suspend states.

- Set the following CAR register bits to '1' to put the PLL to IDDQ.
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE]

The System Power Management driver keeps UTMIPLL's IDDQ control in software control.

- When all USB2 ports are in Disconnected or Suspend states, set the following CAR register bit to '1' to put the PLL to IDDQ:
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE]
- When any USB2 port is exiting Disconnected or Suspend states, clear the following CAR register bit to '0':
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE]

22.8.6.2 Disable Controller Clock Gating

Step 1

The System Power Management driver brings UTMIPLL out of to IDDQ when all non-HSIC USB2.0 ports are assigned to USB2 controllers and any of these controllers are exiting reset or suspend states.

- Set the following CAR register bits to '0' to put the PLL out of IDDQ.
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE]

The System Power Management driver puts UTMIPLL IDDQ to under software control when some, but not all, non-HSIC USB2.0 ports are assigned to USB2 controllers, and any of these controllers are exiting reset or suspend states.

- Set the following CAR register bits to ‘1’ to disable IDDQ hardware control.
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_PD_INCLUDE]
  - CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_SWCTL]

Step 2

The System Power Management driver can be used to enable the USB2 controller clocks.

- Set the following USB2 register bits to ‘1’ to enable the clocks.
  - CLK_RST_CONTROLLER_CLK_ENB_L_SET_0[SET_CLK_ENB_USBD] for USB2 #1
  - CLK_RST_CONTROLLER_CLK_ENB_H_SET_0[SET_CLK_ENB_USB2] for USB2 #2

Step 3

The System Power Management driver disables the USB2 controller wakeup interrupt.

- Set the following USB2 register bits to ‘0’ to disable the interrupt.
  - USB2_UTMIP_PMC_WAKEUP0_0[UTMIP_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #1
  - UHSIC_PMC_WAKEUP0_0[UHSIC_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #2

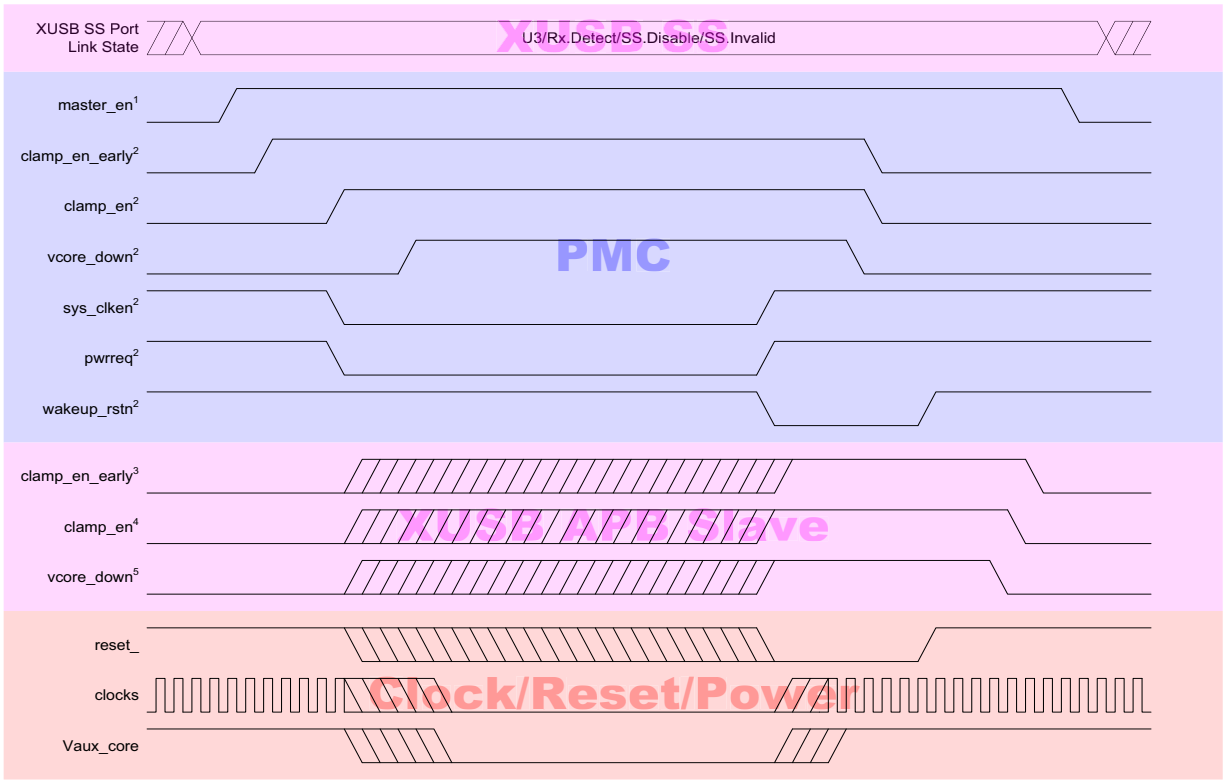
Step 4

The System Power Management driver programs the PMC USB2.0 sleepwalk logic to disable the sleepwalk logic as described in Section 22.8.9.3: Disable PMC Sleepwalk Logic.

22.8.7 XUSB Controller Power Gating

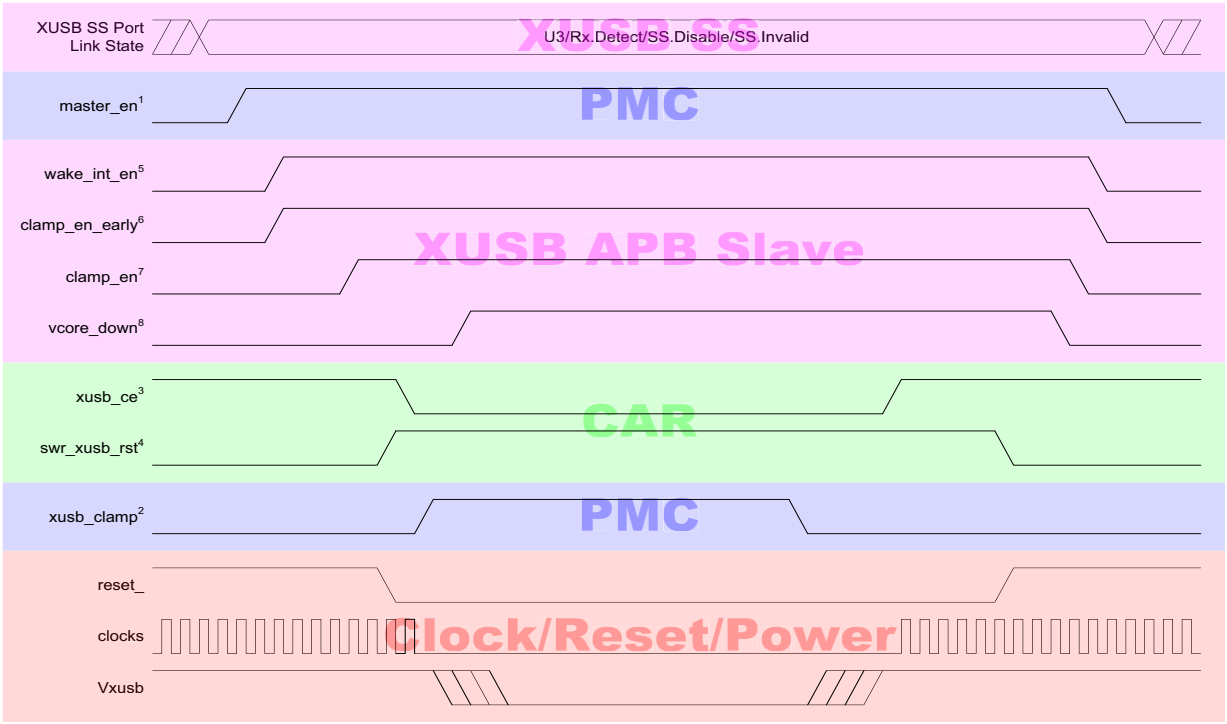
The following figure illustrates the signal sequences from the PMC that are used by XUSB SS Wake logic to enable its wake event detection.

Figure 58: PMC Signals to XUSB SS Wake Logic



The following figure illustrates the signal sequences from the XUSB PADCTL block that are required to follow the same sequence as from the PMC so XUSB SS Wake logic can enable its wake event detection.

Figure 59: XUSB PADCTL Signals to XUSB SS Wake Logic



The XUSB controller has three partitions that can be selectively power gated with the following use cases, where unlisted combinations are not supported:

Table 111: Power Gating of XUSB Partitions

XUSBC (Host/USB2.0)	XUSBB (Device)	XUSBA (SuperSpeed)	Use Case Descriptions
Powered	Powered	Powered	Both Host and Device modes in normal operations, with SuperSpeed links operational.
Powered	Powered	Power Gated	Both Host and Device modes in normal operations, with SuperSpeed links in low power states. SuperSpeed link wake events report through XUSB PADCTL.
Powered	Power Gated	Powered	Host mode in normal operations, with SuperSpeed links operational. Device mode power gated. Device mode SuperSpeed and USB2.0 link wake events report through XUSB PADCTL.
Powered	Power Gated	Power Gated	Host mode in normal operations, with SuperSpeed links in low power states. Device mode power gated. Host Mode SuperSpeed, and Device mode SuperSpeed and USB2.0 link wake events report through XUSB PADCTL.
Power Gated	Powered	Powered	Host mode power gated with USB2.0 buses in low power states, with Device mode and SuperSpeed link in normal operation. Host Mode SuperSpeed and USB2.0 link wake events report through XUSB PADCTL.
Power Gated	Power Gated	Power Gated	Both Host and Device modes power gated, with SuperSpeed links in low power states. All Host and Device modes link wake events report through XUSB PADCTL.

Due to the wake latency requirements, device partition can only be power-gated when the device port is not connected.

22.8.7.1 All Partitions ELPG Entry

Step 1

The xHCI driver performs context save operation as described in Section 4.23.2 of the xHCI specification.

The xHCI PEP driver performs XUSB_HOST specific context save operation. In case the SuperSpeed partition has already been power gated, xHCI FW and XUSB Device Mode drivers should not save the context of the SuperSpeed partition again.

The XUSB Device Mode driver performs XUSB_DEVICE specific context save operations.

The xHCI PEP driver and XUSB Device Mode driver perform XUSB IPFS specific context save operations.

- Read and store the value of the following registers
  - XUSB_{HOST,DEV}_MSI_BAR_SZ_0
  - XUSB_{HOST,DEV}_MSI_AXI_BAR_ST_0
  - XUSB_{HOST,DEV}_MSI_FPCI_BAR_ST_0
  - XUSB_{HOST,DEV}_MSI_VEC0_0
  - XUSB_{HOST,DEV}_MSI_EN_VEC0_0
  - XUSB_{HOST,DEV}_FPCI_ERROR_MASKS_0
  - XUSB_{HOST,DEV}_INTR_MASK_0
  - XUSB_{HOST,DEV}_IPFS_INTR_ENABLE_0
  - XUSB_{HOST,DEV}_UFPCI_CONFIG_0
  - XUSB_{HOST,DEV}_CLKGATE_HYSTERESIS_0
  - XUSB_{HOST,DEV}_XUSB_HOST_MCCIF_FIFOCTRL_0

Step 2

The System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB Host Mode and XUSB Device Mode according as described in the “PMC Programming” section.

The System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

Note: *The line wakeup event from PMC is shared between the XUSB and USB2 controller. There is no per controller masking for the line wakeup event, and when the sleepwalk logic triggers the event, both XUSB and USB2 controllers will log the status even when they are not power gated or clock gated. Thus the system power management driver should ensure the wake status is cleared before enabling the sleepwalk logic.*

Step 3

The xHCI PEP driver and XUSB Device Mode driver enable the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports assigned to host and device.

- Write '1' to the following XUSB PADCTL register bits to clear the interrupt status.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT3_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT2_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT1_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT0_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKEUP_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_HSIC_PORT0_WAKEUP_EVENT]
- Set the following XUSB PADCTL register bits to '1' to enable the interrupt.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT3_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT2_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]

Note: *The interrupt status bit could also be set due to previous wake events reported for USB2 controllers, thus the status should be cleared before enabling the interrupt.*

Step 4

The xHCI PEP driver and XUSB Device Mode driver initiate the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports assigned to host and device.

- Write '1' to the following XUSB PADCTL register bits to assert the clamp_en_early signal.
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN_EARLY]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN_EARLY]

- XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN_EARLY]
- XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN_EARLY]

- Write '1' to the following XUSB PADCTL register bits to assert the clamp_en signal.
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN]
- Wait 250 µs

Note: *These two writes must not be combined.*

Step 5

System Power Management driver flushes MCCIF and partition clients.

- Set the following MC register bits to '1' to enable flush to XUSB
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for Host Mode
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_DEV_FLUSH_ENABLE] for Device Mode
- Read the following MC register bits to be '1' to ensure flush to XUSB is enabled
  - MC_CLIENT_HOTRESET_STATUS_0[XUSB_HOST_HOTRESET_STATUS] for Host Mode
  - MC_CLIENT_HOTRESET_STATUS_0[XUSB_DEV_HOTRESET_STATUS] for Device Mode

Step 6

The System Power Management driver asserts reset to XUSB then disables its clocks.

- Set the following CAR register bits to '1' to assert reset to XUSB
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for Host Mode
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for Device Mode
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports
- Set the following CAR register bits to '1' to disable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_HOST] for Host Mode
  - CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_DEV] for Device Mode
  - CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0[CLR_CLK_ENB_XUSB_SS] for SS ports

Step 7

The System Power Management driver disables the XUSB power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB host:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'enable'
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBC'
- Read the following PMC register bit to confirm the power gating status of XUSB host:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals 'OFF'
- Program the following PMC register bits in a single write to disable the power rail to XUSB device:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'enable'
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBB'

- Read the following PMC register bit to confirm the power gating status of XUSB device:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBB] equals ‘OFF’
- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’
- Read the following PMC register bit to confirm the power gating status of XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘OFF’

Note: *Only one partition can be power gated at a time.*

Step 8

The xHCI PEP driver and XUSB Device Mode driver initiate the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports assigned to host and device.

- Write ‘1’ to the following XUSB PADCTL register bits to assert the vcore_off signal:
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_VCORE_DOWN]

22.8.7.2 Host Mode Partition ELPG Entry

Step 1

The xHCI driver performs context save operation as described in Section 4.23.2 of the xHCI specification.

The xHCI PEP driver performs XUSB_HOST specific context save operations. In case the SuperSpeed partition has already been power gated, xHCI FW should not save the context of the SuperSpeed partition again.

The xHCI PEP driver performs XUSB IPFS specific context save operations.

Step 2

The System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB Host Mode as described in the “PMC Programming” section.

The System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

Note: *The line wakeup event from PMC is shared between XUSB and USB2 controllers. There is no per controller masking for the line wakeup event, and when the sleepwalk logic triggers the event, both XUSB and USB2 controllers will log the status even when they are not power gated or clock gated. Thus the system power management driver should ensure the wake status is cleared before enabling the sleepwalk logic.*

Step 3

The xHCI PEP driver enables the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports assigned to the host.

Step 4

The System Power Management driver flushes MCCIF and partition clients.

- Set the following MC register bit to ‘1’ to enable flush to XUSB:

- MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for Host Mode

- Read the following MC register bit to be ‘1’ to ensure flush to XUSB is enabled:
  - MC_CLIENT_HOTRESET_STATUS_0[XUSB_HOST_HOTRESET_STATUS] for Host Mode

Step 5

The System Power Management driver asserts reset to XUSB then disables its clocks.

- Set the following CAR register bit to ‘1’ to assert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for Host Mode
- Set the following CAR register bit to ‘1’ to disable the clocks to individual XUSB partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_HOST] for Host Mode

Step 6

The System Power Management driver disables the XUSB power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB host:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’
- Read the following PMC register bit to confirm the power gating status of XUSB host:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals ‘OFF’

22.8.7.3 SuperSpeed Partition ELPG Entry

Step 1

The xHCI PEP driver and XUSB Device Mode driver communicate SuperSpeed partition ELPG entry via mailbox protocol.

The xHCI PEP driver and XUSB Device Mode driver communicate SuperSpeed partition ELPG entry with the System Power Management driver.

The xHCI FW and XUSB Device Mode driver perform their SuperSpeed port specific context save operations.

Step 2

The xHCI PEP driver and XUSB Device Mode driver enable the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports assigned to host and device as described in Step 3 of the “All Partitions ELPG Entry” section.

Step 3

The xHCI PEP driver and XUSB Device Mode driver initiate the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed portsassigned to host and device as described in Step 4 of the “All Partitions ELPG Entry” section.

Step 4

The System Power Management driver asserts reset to XUSB SuperSpeed partition then disables its clocks.

- Set the following CAR register bit to ‘1’ to assert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports
- Set the following CAR register bit to ‘1’ to disable the clocks to individual XUSB partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0[CLR_CLK_ENB_XUSB_SS] for SS ports

Step 5

The System Power Management driver disables the XUSB SuperSpeed partition power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed:

- APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
- APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’

- Read the following PMC register bit to confirm the power gating status of the partitions:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘OFF’

Note: *Only one partition can be power gated at a time.*

Step 6

The xHCI PEP driver and XUSB Device Mode driver initiate the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports assigned to host and device as described in Step 7 of the “All Partitions ELPG Entry” section.

22.8.7.4 Device Mode Partition ELPG Entry

Device mode partition should only be power gated when the device port is not connected.

Step 1

The XUSB Device Mode driver performs XUSB_DEVICE specific context save operations.

The XUSB Device Mode driver performs XUSB IPFS specific context save operation as described in step 1 of the “All Partitions ELPG Entry” section.

Step 2

The System Power Management driver programs the PMC to enable wake on connect where connection event is triggered by valid VBUS.

Step 3

The System Power Management driver flushes MCCIF and partition clients.

- Set the following MC register bit to ‘1’ to enable flush to XUSB:
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_DEV_FLUSH_ENABLE] for Device Mode
- Read the following MC register bit to be ‘1’ to ensure flush to XUSB is enabled:
  - MC_CLIENT_HOTRESET_STATUS_0[XUSB_DEV_HOTRESET_STATUS] for Device Mode

Step 4

The System Power Management driver asserts reset to XUSB then disables its clocks.

- Set the following CAR register bit to ‘1’ to assert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for Device Mode
- Set the following CAR register bit to ‘1’ to disable the clocks to individual XUSB partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_DEV] for Device Mode

Step 5

The System Power Management driver disables the XUSB power rails.

- Programming the following PMC register bits in a single write to disable the power rail to XUSB device:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBB’
- Read the following PMC register bit to confirm the power gating status of the partitions:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBB] equals ‘OFF’

Note: *Only one partition can be power gated at a time.*

Step 6

The XUSB Device Mode driver initiates the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports assigned to device as described in step 7 of the “All Partitions ELPG Entry” section.

- Write ‘1’ to the following XUSB PADCTL register bit to assert the vcore_off signal.
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_VCORE_DOWN]

22.8.7.5 All Partitions ELPG Exit

In the case the wake event is not generated by SuperSpeed ports, the SuperSpeed related programming should be skipped, where the SuperSpeed partition should exit ELPG due to wake events detected by SuperSpeed ports.

Step 1

The xHCI PEP driver and XUSB Device Mode driver move the USB2.0 port back to XUSB.

- Program the following XUSB PADCTL registers to assign the USB2.0 ports back to XUSB, according to the platform specific configuration:
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT3]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT2]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT1]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT0]

Step 2

The System Power Management driver enables the XUSB power rails.

- Program the following PMC register bits in a single write to enable the power rail to XUSB host:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’
- Read the following PMC register bit to confirm the power gating status of XUSB host:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals ‘ON’
- Program the following PMC register bits in a single write to disable the power rail to XUSB device:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBB’
- Read the following PMC register bit to confirm the power gating status of XUSB device:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBB] equals ‘ON’
- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’
- Read the following PMC register bit to confirm the power gating status of XUSB SuperSpeed:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘ON’

Note: *Only one partition can be un-power gated at a time.*

Step 3

The System Power Management driver enables XUSB clocks.

- Set the following CAR register bits to '1' to enable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_HOST] for Host Mode
  - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_DEV] for Device Mode
  - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS] for SS ports

Step 4

The System Power Management driver removes power clamps to XUSB partitions.

- Set the following PMC register bits to '1' to remove the power clamps to individual XUSB partitions.
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] for Host Mode
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBB] for Device Mode
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA] for SS ports
- Read the following PMC register bits to confirm the power clamps to individual XUSB partitions are removed.
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] equals '0'
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBB] equals '0'
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA] equals '0'

Step 5

The System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bits to '0' to deassert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for Host Mode
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for Device Mode
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports
- Wait 1 µs

Step 6

The System Power Management driver disables flushes of MCCIF and partition clients.

- Set the following MC register bits to '0' to enable flush to XUSB:
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for Host Mode
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_DEV_FLUSH_ENABLE] for Device Mode

Step 7

The xHCI PEP driver and XUSB Device Mode driver disable the XUSB wakeup interrupts.

- Set the following XUSB PADCTL register bits to '0' to disable the interrupts.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT3_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT2_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]

- XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKE_INTERRUPT_ENABLE]
- XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKE_INTERRUPT_ENABLE]
- XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]
- XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]
- XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]

Step 8

The xHCI PEP driver and XUSB Device Mode driver initiate the signal sequence to disable the XUSB SS wake detection logic.

- Write ‘0’ to the following XUSB PADCTL register bits to deassert the vcore_off signal.
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_VCORE_DOWN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_VCORE_DOWN]
- Wait 100 µs.
- Write ‘0’ to the following XUSB PADCTL register bits to deassert the clamp_en_early signals.
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN_EARLY]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN_EARLY]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN_EARLY]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN_EARLY]
- Wait 100 µs.
- Write ‘0’ to the following XUSB PADCTL register bits to deassert the clamp_en signals.
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP3_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP2_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP1_ELPG_CLAMP_EN]
  - XUSB_PADCTL_ELPG_PROGRAM_1_0[SSP0_ELPG_CLAMP_EN]

Note: *The write to clear vcore_off cannot be combined with the write to clear clamp_en and clamp_en_early.*

Step 9

The xHCI PEP driver performs XUSB IPFS and XUSB register initialization.

The xHCI PEP driver and XUSB Device Mode driver perform XUSB IPFS specific context restore operations for registers listed in Step 1 of the “All Partitions ELPG Entry” section.

The xHCI PEP driver performs XUSB context restore operation.

The XUSB Device Mode driver performs XUSB_DEVICE context restore operations.

Step 10

The xHCI PEP driver performs XUSB HSIC specific context restore operations by reading the wake status of the HSIC ports from PADCTL registers and programming the status to XUSB registers.

- If XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKEUP_EVENT] is ‘1’, set the following XUSB register:
  - XUSB_CFG_ARU_CONTEXT_HS_PLS _PORT4 to ‘RESUME’

Else if XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKEUP_EVENT] is ‘0’, set the following XUSB register:

- XUSB_CFG_ARU_CONTEXT_HS_PLS _PORT4 to ‘SUSPEND’

- If XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKEUP_EVENT] is ‘1’, set the following XUSB register:
  - XUSB_CFG_ARU_CONTEXT_HS_PLS _PORT5 to ‘RESUME’

    Else if XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKEUP_EVENT] is ‘0’, set the following XUSB register:
  - XUSB_CFG_ARU_CONTEXT_HS_PLS _PORT5 to ‘SUSPEND’

Step 11

The xHCI PEP driver loads XUSB firmware.

The xHCI PEP driver notifies XUSB firmware whether context of SuperSpeed Partition should be restored.

Step 12

The xHCI driver performs context restore operations as described in Section 4.23.2 of the xHCI specification.

Step 13

The System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB Host Mode and XUSB Device Mode to disable the sleepwalk logic as described in the “PMC Programming” section.

The xHCI PEP driver and XUSB Device Mode driver clear the XUSB wakeup events.

- Write ‘1’ to the following XUSB PADCTL register bits to clear the events.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT3_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT2_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[SS_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_HSIC_PORT0_WAKE_EVENT]

22.8.7.6 Host Mode ELPG Exit

Step 1

The xHCI PEP driver and XUSB Device Mode driver moves the USB2.0 port back to XUSB.

- Program the following XUSB PADCTL registers to assign the USB2.0 ports to XUSB, according to the platform specific configuration:
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT3]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT2]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT1]
  - XUSB_PADCTL_USB2_PAD_MUX_0[USB2_OTG_PAD_PORT0]

Step 2

The System Power Management driver enables the XUSB power rails.

- Program the following PMC register bits in a single write to enable the power rail to XUSB host:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’
- Read the following PMC register bit to confirm the power gating status of XUSB host:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBC] equals ‘ON’

Step 3

The System Power Management driver enables XUSB clocks.

- Set the following CAR register bit to ‘1’ to enable the clocks to individual XUSB partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_HOST] for Host Mode

Step 4

The System Power Management driver removes power clamps to XUSB partitions.

- Set the following PMC register bit to ‘1’ to remove the power clamps to individual XUSB partitions:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] for Host Mode
- Read the following PMC register bit to confirm the power clamps to individual XUSB partitions are removed:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBC] equals ‘0’

Step 5

The System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bit to ‘0’ to deassert reset to XUSB
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for Host Mode
- Wait 1 µs

Step 6

The System Power Management driver disables flushes of MCCIF and partition clients.

- Set the following MC register bits to ‘0’ to disable flush to XUSB
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_HOST_FLUSH_ENABLE] for Host Mode

Step 7

The xHCI PEP driver disables the USB2.0 wakeup interrupts for ports assigned to XUSB.

- Set the following XUSB PADCTL register bits to ‘0’ to disable the interrupts.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]

Step 8

The xHCI PEP driver performs XUSB IPFS and XUSB register initialization.

The xHCI PEP driver performs XUSB IPFS context restore operation.

The xHCI PEP driver performs XUSB context restore operations.

Step 9

The xHCI PEP Driver performs HSIC context restore operations as described in Step 10 of the “All Partitions ELPG Exit” section.

The xHCI PEP Driver loads XUSB firmware.

The xHCI PEP Driver notifies XUSB firmware whether context of SuperSpeed Partition should be restored.

Step 10

The xHCI driver performs context restore operation as described in Section 4.23.2 of the xHCI specification.

Step 11

The System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB Host Mode to disable the sleepwalk logic as described in the “PMC Programming” section.

The xHCI PEP driver clears the XUSB wakeup events.

- Write ‘1’ to the following XUSB PADCTL register bits to clear the events.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_HSIC_PORT0_WAKE_EVENT]

22.8.7.7 SuperSpeed Partition ELPG Exit

Step 1

The xHCI FW and xHCI PEP drivers communicate SuperSpeed partition ELPG exit via mailbox protocol.

The xHCI PEP driver and XUSB Device Mode driver communicate SuperSpeed partition ELPG exit with the System Power Management driver.

Step 2

The System Power Management driver enables the XUSB power rails.

- Program the following PMC register bits in a single write to disable the power rail to XUSB SuperSpeed
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’
- Read the following PMC register bits to confirm the power gating status of XUSB SuperSpeed
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBA] equals ‘ON’

Note: *Only one partition can be un-power gated at a time*

Step 3

The System Power Management driver enables XUSB clocks.

- Set the following CAR register bit to ‘1’ to enable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS] for SS ports

Step 4

The xHCI PEP driver and XUSB Device Mode driver initiate the signal sequence to disable the XUSB SS wake detection logic for the SuperSpeed ports assigned to host and device as described in Step 3 of the “All Partitions ELPG Exit” section.

Step 5

The xHCI PEP driver and XUSB Device Mode driver disable the XUSB SS wakeup interrupts.

- Set the following XUSB PADCTL register bits to '0' to disable the interrupts.
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]
  - XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]

Step 6

The System Power Management driver removes power clamps to XUSB partitions.

- Set the following PMC register bit to '1' to remove the power clamp to XUSB SS:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA]
- Read the following PMC register bits to confirm the power clamp to XUSB SS is removed.
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBA] equals '0'

Step 7

The System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bit to '0' to deassert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports

Step 8

The System Power Management driver disables flushes of MCCIF and partition clients.

- Set the following MC register bit to '0' to disable flush to XUSB
  - MC_CLIENT_HOTRESET_CTRL_0[XUSB_DEV_FLUSH_ENABLE] for Device Mode

Step 9

The xHCI FW and XUSB Device Mode drivers perform their SuperSpeed port context restore operations.

The xHCI PEP driver and XUSB Device Mode driver clear the XUSB wakeup events.

- Write '1' to the following XUSB PADCTL register bits to clear the events.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT0_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_HSIC_PORT0_WAKE_EVENT]

22.8.7.8 Device Mode Partition ELPG Exit

Step 1

The System Power Management driver enables the XUSB power rails.

- Programming the following PMC register bits in a single write to disable the power rail to the XUSB device:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'enable'
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBB'
- Read the following PMC register bit to confirm the power gating status of the XUSB device:
  - APBDEV_PMC_PWRGATE_STATUS_0[XUSBB] equals 'ON'

Note: *Only one partition can be un-power gated at a time*

Step 2

The System Power Management driver enables XUSB clocks.

- Set the following CAR register bit to '1' to enable the clocks to individual XUSB partitions.
  - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_DEV] for Device Mode

Step 3

The XUSB Device Mode driver initiates the signal sequence to disable the *XUSB* SS wake detection logic for the SuperSpeed ports assigned to the device as described in Step 3 of the "All Partitions ELPG Exit" section.

Step 4

XUSB Device Mode driver disables the *XUSB* SS wakeup interrupts for the ports assigned to device as described in Step 4 of the "All Partitions ELPG Exit" section.

Step 5

The System Power Management driver removes power clamps to XUSB partitions:

- Set the following PMC register bit to '1' to remove the power clamp to the XUSB device:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBB]
- Read the following PMC register bit to confirm the power clamp to the XUSB device is removed:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [XUSBB] equals '0'

Step 6

The System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bit to '0' to deassert reset to XUSB:
  - CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for Device Mode

Step 7

The System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB Device Mode to disable the sleepwalk logic as described in the "Disable PMC Sleepwalk Logic" section.

The XUSB Device Mode driver clears the XUSB wakeup events:

- Write '1' to the following XUSB PADCTL register bits to clear the events.
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT3_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT2_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT1_WAKE_EVENT]
  - XUSB_PADCTL_ELPG_PROGRAM_0_0[USB2_PORT0_WAKE_EVENT]

Step 8

The XUSB Device Mode driver performs XUSB IPFS context restore operations for registers listed in Step 1 of "Cold Boot" then enables XUSB IPFS operation.

- Set the following XUSB IPFS register bit to '1':
  - XUSB_DEV_CONFIGURATION_0[EN_FPCI]

The XUSB Device Mode driver performs XUSB_DEVICE context restore operations.

Before power is to be removed from USB3, software should set the Device Mode to stop, then read the contexts from Device Mode MMIO registers and put them in system memory if required. After power is restored to USB3 and hardware reset to USB3 is deasserted, software should pull the contexts from system memory and write them to the Device Mode registers. Software should reprogram the Device Mode registers to re-initialize the data structures and transfer rings before setting the Device Mode to run.

The following Device Mode contexts should be saved and restored by the driver in Linux systems and are assumed to be saved and restored by the OS in Windows systems – if this is not the case, the Windows PEP filter driver will have to perform the context save and restore:

- PCI config registers
  - Command and Status
  - BAR0/1
  - Interrupt Line
  - FLADJ
  - MSI Message Control
  - MSI Address
  - MSI Data
  - Power Management Control and Status

The following Device Mode contexts should be saved and restored by the Device Mode Driver:

- MMIO registers:
  - Device Address
  - U2 Timeout
  - Port Link State and Port State
  - Event Ring enqueue pointer and PCS

22.8.8 XUSB HSIC Programming

22.8.8.1 HSIC Power Up (Airplane Mode Exit)

The default state of pull up and pull down for data and strobe for HSIC is pull down enabled on both Data and Strobe. During power up of the chip, the pull down maintains a reset state on the bus and the device waits for idle on the bus before signaling a connect event. Exit from airplane mode is based on mailbox message from software.

During power up, xHCI PEP driver must:

- enable the power supply for HSIC power rail of Tegra and the external devices
- load firmware
- direct the PSM to disconnect state
- clear CNR
- initiate a mailbox to enable PU/PD configuration to maintain IDLE state as explained in “HSIC Power Down (Airplane Mode Entry)”.

Note: *PLLU must be powered up to detect a connect event. So until all HSIC devices are detected, PLL power down must be disabled.*

Once a connect event is detected and software issues a port reset on the port, the HSIC PSM moves into U0. xHCI FW generates a mailbox to remove the PU/PD on the HSIC port, and after that generates a port status change event for port

reaching U0. The idea is to prevent device enumeration unless PU/PD is removed after going to U0 since it will impact both power and electrical. Note that PLL shutdown can be enabled at this point.

22.8.8.2 HSIC Power Down (Airplane Mode Entry)

In Airplane mode, xHCI PEP Driver must restore the PU/PD to drive pull down on both data and strobe. It also informs FW to direct the port to disconnect state and generate a port status change event to software signaling a disconnect.

22.8.8.3 LP0 Entry/Exit

During LP0, the PADCTL block is under reset. So during exit, it assumes the power on reset value which is pull down on both data/strobe. During LP0 exit, either host initiated / device initiated, after restoring the PSM state to U3/resume, the xHCI PEP driver removes the PU/PD on the HSIC port and transfers control to the controller from PMC ensuring a smooth transition.

22.8.8.4 ELPG Entry/Exit

In ELPG, the state of the PADCTL block is maintained, which is no PU/PD for data and strobe. Once the PSM state is restored to U3/resume, control is transferred from the PMC to XUSB, ensuring a smooth transition.

22.8.8.5 HSIC Pull-Up/Pull-Down Resistors Programming

The xHCI PEP Driver sets the following XUSB PADCTL registers in response to mailbox requests from xHCI FW to disable the HSIC pull-up/pull-down resistors of HSIC port 0.

- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPU_STROBE] to ‘0’
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPU_DATA0] to ‘0’
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPD_STROBE] to ‘0’
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPD_DATA0] to ‘0’

The xHCI PEP Driver sets the following XUSB PADCTL registers in response to mailbox requests from xHCI FW to enable the HSIC pull-up/pull-down resistors of HSIC port 0 to keep the bus in IDLE state.

- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPU_STROBE] to ‘1’
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPU_DATA0] to ‘0’
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPD_STROBE] to ‘0’
- XUSB_PADCTL_HSIC_PAD0_CTL_0_0[RPD_DATA0] to ‘1’

22.8.8.6 HSIC Device Reset Toggling/Power Cycling Due to XUSB Host Controller Reset (HCRST)

When the mailbox request message is received from XUSB FW for resetting the HSIC device, the PEP Filter driver should communicate with the system power management driver to either assert then deassert the reset to the HSIC devices or remove then enable the power rail to the HSIC device through platform specific means, and only acknowledge the mailbox request after the HSIC device has been reset.

22.8.8.7 HSIC Device Power Down

The HSIC does not define methods to disconnect and power off a link. The following sequences allow powering down of the HSIC device for use cases such as powering down HSIC modems when enabling flight mode.

Step 1

The xHCI Driver stops all Endpoints of the HSIC device then disables the slot assigned to the HSIC device.

Step 2

The xHCI Driver sets the following bit to put the HSIC port to POWERED OFF state:

- XUSB_XHCI_OP_PORTSC_PP to ‘0’

Step 3

The System Power Management driver asserts the reset to the HSIC device then disables the power rail to the HSIC device.

22.8.8.8 HSIC Device Power Up

The following sequences allow powering up of the HSIC device for use cases such as powering up HSIC modems when disabling flight mode.

Step 1

The xHCI Driver sets the following bit to put the HSIC port to the DISCONNECT state:

- XUSB_XHCI_OP_PORTSC_PP to ‘1’

Step 2

The System Power Management driver enables the power rail to the HSIC device then deasserts the reset to the HSIC device.

Step 3

The xHCI Driver sets the following bit to put the HSIC port to the ENABLED state:

- XUSB_XHCI_OP_PORTSC_PR to ‘1’

Step 4

The xHCI Driver enumerates the HSIC device.

22.8.8.9 Mailbox Details

The message names are listed below. For all messages, the port number is bit-mapped. If it is port 1, bit 1 will be set.

- SW_FW_MBOX_CMD_AIRPLANE_MODE_DISABLED
- SW_FW_MBOX_CMD_AIRPLANE_MODE_ENABLED
- SW_FW_MBOX_CMD_ENABLE_PUPD
- SW_FW_MBOX_CMD_DISABLE_PUPD

22.8.9 PMC Programming

Sleepwalk logic in the PMC is used for wake event detection of USB2.0 ports, including wake on connect, wake on disconnect, and remote wakeup for both USB2 and XUSB.

22.8.9.1 Initialize PMC Sleepwalk Logic

Step 1

The xHCI PEP driver initializes the sleepwalk logic:

- Set the following registers to ‘0’ to ensure sleepwalk logic is disabled:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_MASTER_ENABLE_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
- Set the following registers to ‘1’ to ensure sleepwalk logic is in low power mode:
  - APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P0]
  - APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P1]

- APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P2]
- APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UTMIP_PWR_P3]
- APBDEV_PMC_UTMIP_MASTER_CONFIG_0[UHSIC_PWR_P0]

- Program the following registers to set debounce time:
  - APBDEV_PMC_USB_DEBOUNCE_DEL_0[UTMIP_LINE_DEB_CNT]
  - APBDEV_PMC_USB_DEBOUNCE_DEL_0[UHSIC_LINE_DEB_CNT]
- Set the following registers to '0' to ensure fake events of sleepwalk logic are disabled:
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_VAL_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_VAL_P1]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBOP_VAL_P2]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UTMIP_FAKE_USBON_VAL_P2]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBOP_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBON_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBOP_VAL_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_FAKE_0[UTMIP_FAKE_USBON_VAL_P3]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_DATA_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_STROBE_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_DATA_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_FAKE_0[UHSIC_FAKE_STROBE_VAL_P0]
- Set the following registers to '0' to ensure wake events of sleepwalk logic are not latched:
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UHSIC_LINE_WAKEUP_EN_P0]
- Program the following registers to 'NONE' to disable wake event triggers of sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_WAKE_VAL_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_WAKE_VAL_P0]

- Set the following registers to ‘1’ to power down the line state detectors of the pad:
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[DATA0_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[DATA1_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[STROBE_VAL_PD_P0]

22.8.9.2 Enable PMC Sleepwalk Logic

Step 1

The xHCI PEP driver re-initializes the sleepwalk logic as described in the “Initialize PMC Sleepwalk Logic” section.

Step 2

The xHCI PEP driver sets up the sleepwalk logic.

- Program the following registers to match the speed of the port:
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UTMIP_SPEED_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UTMIP_SPEED_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UTMIP_SPEED_P2]
  - APBDEV_PMC_UTMIP_UHSIC2_SAVED_STATE_0[UTMIP_SPEED_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0[UHSIC_SPEED_P0]
- Set the following registers to ‘1’ to enable the trigger of the sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC2_SLEEPWALK_CFG_0[UTMIP_WAKE_WALK_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC2_SLEEPWALK_CFG_0[UTMIP_LINEVAL_WALK_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UHSIC_WAKE_WALK_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0[UHSIC_LINEVAL_WALK_EN_P0]
- Write ‘1’ to the following registers to reset the walk pointer and clear the alarm of the sleepwalk logic, as well as capture the configuration of the USB2.0 pad:
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P0]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P0]

- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P0]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P1]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P1]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P1]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P2]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P2]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P2]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WALK_PTR_P3]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P3]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CAP_CFG_P3]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UHSIC_CLR_WALK_PTR_P0]
- APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UHSIC_CLR_WAKE_ALARM_P0]

- Program the following registers with electrical parameters read from XUSB PADCTL:
  - APBDEV_PMC_UTMIP_TERM_PAD_CFG_0[TCTRL_VAL] to XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0[TCTRL]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[PCTRL_VAL] to XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0[PCTRL]
- Program the following registers to set up the pull-ups and pull-downs of the signals during the four stages of sleepwalk:
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBOP_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBON_RPD_{A/B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBOP_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[USBON_RPU_{A/B/C/D}] to ‘0’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPD_A] to ‘0’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPD_A] to ‘1’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPD_A] to ‘1’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPU_A] to ‘1’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPU_A] to ‘0’
  - APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPU_A] to ‘0’

- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPD_{B/C/D}] to ‘1’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPD_{B/C/D}] to ‘0’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPD_{B/C/D}] to ‘1’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[STROBE_RPU_{B/C/D}] to ‘0’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA0_RPU_{B/C/D}] to ‘1’
- APBDEV_PMC_UHSIC_SLEEPWALK_P0_0[DATA1_RPU_{B/C/D}] to ‘0’

- Program the following registers to set up the driving values of the signals during the four stages of sleepwalk:
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P0_0[HIGHZ_{B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P1_0[HIGHZ_{B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P2_0[HIGHZ_{B/C/D}] to ‘1’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AP_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AN_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[HIGNZ_A] to ‘0’
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AP_{B/C/D}] to ‘0’ for HS/FS and ‘1’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[AN_{B/C/D}] to ‘1’ for HS/FS and ‘0’ for LS
  - APBDEV_PMC_UTMIP_SLEEPWALK_P3_0[HIGHZ_{B/C/D}] to ‘1’

Note: *For HSIC, DATA1 is not used for signaling wake.*

Step 3

The xHCI driver puts the ports in the U3 suspend state.

Step 4

The xHCI PEP driver enables the wake event detections.

- Set the following registers to ‘0’ to power up the line state detectors of the pad:

- APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P0]
- APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P0]
- APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P1]
- APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P1]
- APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P2]
- APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P2]
- APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P3]
- APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P3]
- APBDEV_PMC_USB_AO_0[DATA0_VAL_PD_P0]
- APBDEV_PMC_USB_AO_0[DATA1_VAL_PD_P0]
- APBDEV_PMC_USB_AO_0[STROBE_VAL_PD_P0]

- Wait 1 µs.
- Set the following registers to ‘1’ to switch the electric control of the USB2.0 pad to PMC:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_FSLS_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_PCTRL_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_TCTRL_USE_PMC_P3]
- Program the following registers to set the wake signaling trigger events
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P0] to ‘ANY’
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P1] to ‘ANY’
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P2] to ‘ANY’
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_WAKE_VAL_P3] to ‘ANY’
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_WAKE_VAL_P0] to ‘SD10’
- Set the following registers to ‘1’ to enable the wake detection
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P1]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_MASTER_ENABLE_P3]

- APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P3]
- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
- APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UHSIC_LINE_WAKEUP_EN_P0]

Notes:

- *For USB2 ports, setting wake value to ANY means any line state change would trigger wake.*
- *For HSIC, DATA1 is not used for wake detection.*

22.8.9.3 Disable PMC Sleepwalk Logic

Step 1

The xHCI PEP driver disables the wake event detections.

- Set the following registers to '0' to disable the wake detection:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P1]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_MASTER_ENABLE_P3]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UTMIP_LINE_WAKEUP_EN_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
  - APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0[UHSIC_LINE_WAKEUP_EN_P0]
- Set the following registers to '0' to switch the electric control of the USB2.0 pad to XUSB or USB2:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P1]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_FSLS_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_PCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_TCTRL_USE_PMC_P2]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_FSLS_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_PCTRL_USE_PMC_P3]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_TCTRL_USE_PMC_P3]
- Program the following registers to 'NONE' to disable wake event triggers of sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P0]
  - APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P1]

- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_WAKE_VAL_P2]
- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG1_0[UTMIP_WAKE_VAL_P3]
- APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_WAKE_VAL_P0]

- Set the following registers to '1' to power down the line state detectors of the pad:
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P1]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P2]
  - APBDEV_PMC_USB_AO_0[USBOP_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[USBON_VAL_PD_P3]
  - APBDEV_PMC_USB_AO_0[DATA0_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[DATA1_VAL_PD_P0]
  - APBDEV_PMC_USB_AO_0[STROBE_VAL_PD_P0]
- Write '1' to the following registers to clear alarm of the sleepwalk logic:
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P0]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P1]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P2]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UTMIP_CLR_WAKE_ALARM_P3]
  - APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0[UHSIC_CLR_WAKE_ALARM_P0]

22.8.10 USB2 Pad Tracking Programming

USB2 pad tracking logic should be triggered once during cold boot and LP0 exit to measure and capture the electric parameters of USB2.0 and HSIC pads.

Step 1

Sets up and enables the tracking clocks.

- Set the following CAR register bits to '1' to enable the tracking clocks
  - CLK_RST_CONTROLLER_CLK_ENB_Y_SET_0[SET_CLK_ENB_USB2_TRK]
  - CLK_RST_CONTROLLER_CLK_ENB_Y_SET_0[SET_CLK_ENB_HSIC_TRK]
- Program the following CAR register bits to set the divisor of tracking clocks
  - CLK_RST_CONTROLLER_CLK_SOURCE_USB2_HSIC_TRK_0[USB2_HSIC_TRK_CLK_DIVISOR]

Note: *The frequency for the tracking circuit should be between 1 to 10 MHz. So if the osc_clk frequency is between 10 to 20 MHz, the clock divisor should be set to 0x2; if the osc_clk frequency is between 20 to 30 MHz, the clock divisor should be set to 0x4; and if the osc_clk frequency is between 30 to 40 MHz, the clock divisor should be set to 0x6.*

Step 2

Check the following registers for the ownership of USB2 BIAS pad and HSIC tracking logic:

- Read the following XUSB registers:

- XUSB_PADCTL_USB2_PAD_MUX_0[USB2_BIAS_PAD]
- XUSB_PADCTL_USB2_PAD_MUX_0[HSIC_PAD_TRK]

Set up the timing parameters:

- Program the following USB2 registers if the pads are assigned to the USB2 controllers:
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_BIAS_TRK_START_COUNT] to ‘0x1E’
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_BIAS_PDTRK_COUNT] to ‘0xA’
  - USB2_UHSIC_PADS_CFG1_0[UHSIC_TRK_START_COUNT] to ‘0x1E’
- Program the following XUSB registers if the pads are assigned to the XUSB controller:
  - XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0[TRK_START_TIMER] to ‘0x1E’
  - XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0[TRK_DONE_RESET_TIMER] to ‘0xA’
  - XUSB_PADCTL_HSIC_PAD_TRK_CTL_0[TRK_START_TIMER] to ‘0x1E’
  - XUSB_PADCTL_HSIC_PAD_TRK_CTL_0[TRK_DONE_RESET_TIMER] to ‘0xA’

Step 3

Power up the pads and the tracking circuit if the pads are assigned to USB2 controllers:

- Set the following USB2 register bits to power up the pads:
  - USB1_UTMIP_BIAS_CFG0_0[UTMIP_BIASPD] to ‘0’
  - USB2_UHSIC_PADS_CFG1_0[UHSIC_PD_TX] to ‘0’
- Wait 1 us
- Set the following USB2 register bits to power up the tracking circuits and enable the hardware state machine:
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_FORCE_PDTRK_POWERUP] to ‘1’
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_FORCE_PDTRK_POWERDOWN] to ‘0’
  - USB2_UHSIC_PADS_CFG1_0[UHSIC_PD_TRK] to ‘0’
- USB2 port has a bug that requires the tracking circuit to be disabled and re-enabled via programming the following USB2 register.
  - Wait 100 us
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_FORCE_PDTRK_POWERDOWN] to ‘1’
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_BIAS_TRK_DONE] to ‘0’
  - Wait 1 us
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_FORCE_PDTRK_POWERDOWN] to ‘0’
  - Wait 100 us
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_FORCE_PDTRK_POWERDOWN] to ‘1’
  - USB1_UTMIP_BIAS_CFG1_0[UTMIP_BIAS_TRK_DONE] to ‘0’
- HSIC port does not have a hardware state machine to automatically disable the tracking, so software should power down the tracking circuits in the following sequence:
  - Read USB2_UHSIC_PADS_CFG1_0[UHSIC_TRK_DONE] equals ‘1’
  - Wait 100 us
  - Set USB2_UHSIC_PADS_CFG1_0[UHSIC_PD_TRK] to ‘1’

Power up the pads and the tracking circuit if the pads are assigned to XUSB controllers:

- Set the following XUSB register bits to power up the pads:

- XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0[PD] to ‘0’
- XUSB_PADCTL_HSIC_PAD1_CTL_0_0[PD_TX_DATA0] to ‘0’
- XUSB_PADCTL_HSIC_PAD1_CTL_0_0[PD_TX_DATA1] to ‘0’
- XUSB_PADCTL_HSIC_PAD1_CTL_0_0[PD_TX_STROBE] to ‘0’

- Wait 1 µs
- Set the following XUSB register bits to power up the tracking circuits and enable the hardware state machine:
  - XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0[PD_TRK] to ‘0’
  - XUSB_PADCTL_HSIC_PAD_TRK_CTL_0[PD_TRK] to ‘0’
  - Wait 100 us

Step 4

Disable the tracking clocks after tracking completed.

- Set the following CAR register bits to ‘1’ to disable the tracking clocks:
  - CLK_RST_CONTROLLER_CLK_ENB_Y_CLR_0[CLR_CLK_ENB_USB2_TRK]
  - CLK_RST_CONTROLLER_CLK_ENB_Y_CLR_0[CLR_CLK_ENB_HSIC_TRK]

22.8.11 Charger Detection 1 (Non-compliant Chargers)

There are four conditions for the different kind of non-compliant chargers: SE1, FS-PU, LS-PU, SE0. Any kind of non-compliant charger can be supported by following the generic procedure below:

1. VBUS detection – by whatever means software does it.
2. Set DIV_DET_EN=1 in XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0.
3. Wait about 10 µs to allow divider detector to settle.
4. Read the {VOP_DIV2P7_DET, VOP_DIV2P0_DET, VON_DIV2P7_DET, VON_DIV2P0_DET} registers bits in XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0.

   {VOP_DIV2P7_DET, VON_DIV2P7_DET} = 2'b11: We are connected to charger_A1.

   {VOP_DIV2P7_DET, VON_DIV2P0_DET} = 2'b11: We are connected to charger_A2.

   {VOP_DIV2P0_DET, VON_DIV2P7_DET} = 2'b11: We are connected to charger_A3.

   {VOP_DIV2P0_DET, VON_DIV2P0_DET} = 2'b11: We are connected to charger_A4.
5. Set DIV_DET_EN=0 in XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0.
6. For A1, A2, A3 or A4, enable charging current as per that charger requirement.
7. Read the {ZIP,ZIN} bits {18, 22} in the XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0 register.

   {ZIP,ZIN} = 2'b11: We are connected to charger_B1.

   {ZIP,ZIN} = 2'b01: We are connected to charger_B2.

   {ZIP,ZIN} = 2'b10: We are connected to charger_B3.

   {ZIP,ZIN} = 2'b00: We are connected to either charger_B4, compliant charger, or a PC host.
8. For B1, B2, or B3, enable charging current as per that charger requirement. This could be different for every implementation.
9. For B4, either we are connected to PC host or a charger_B4 (which could be USB compliant charger as well). Go to Step 10.
10. Enable pull ups on DP and DN.
11. Set XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0 [USBON_RPU_OVRD_VAL]

12. Set XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0 [USBOP_RPU_OVRD_VAL]
13. Read the {ZIP,ZIN} bits {18, 22} in the XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0 register

    {ZIP,ZIN} = 2'b11: We are connected to charger_B1.

    {ZIP,ZIN} = 2'b01: We are connected to charger_B2.

    {ZIP,ZIN} = 2'b10: We are connected to charger_B3.

    {ZIP,ZIN} = 2'b00: We are connected to either charger_B4, compliant charger, or a PC host.
14. Disable pull ups on DP and DN.
15. Clear XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0 [USBON_RPU_OVRD_VAL].
16. Clear XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0 [USBOP_RPU_OVRD_VAL].
17. For B1, B2, or B3, enable charging current as per that charger requirement. This could be different for every implementation.
18. For case B4, either we are connected to PC host or a charger_B4 (which could be USB compliant charger as well).

Charger Detection 2 (Compliant Chargers)

This substitutes step 10 in the above.

1. Make sure the UTMIP_PD_CHRG bit in the UTMIP_BAT_CHRG_CFG0_0 register is set to 0, so that charger detection circuit is on.
2. Set the UTMIP_OP_I_SRC_EN bit in the UTMIP_BAT_CHRG_CFG0_0 register to 1.
3. Wait for about 10 microseconds to allow the battery charger detector to settle.
4. Now check that the VDCD_DET_CHG_DET bit is 1, and the VDCD_DET_STS bit is 0 in the USB_PHY_VBUS_WAKEUP_ID register. This ensures that the data pins have made contact.
5. Set the UTMIP_OP_I_SRC_EN bit to 0 in the UTMIP_BAT_CHRG_CFG0_0 register.
6. Set the following register bits in the UTMIP_BAT_CHRG_CFG0_0 register to:

    UTMIP_OP_SRC_EN = 1

    UTMIP_ON_SINK_EN = 1

    UTMIP_OP_SINK_EN = 0

    UTMIP_ON_SRC_EN = 0
7. Wait for about 10 microseconds to allow the battery charger detector to settle.
8. Now check that the VDAT_DET_CHG_DET and VDAT_DET_STS bits in the USB_PHY_VBUS_WAKEUP_ID register are both set to 1. This indicates connection to a USB compliant dedicated charger that supports up to 1.5A current. If these bits are not set, then the connection is to a PC host, where the downstream port may support charging.
9. Set the following bits in the UTMIP_BAT_CHRG_CFG0_0 register to:

    UTMIP_OP_SRC_EN = 0

    UTMIP_ON_SINK_EN = 0

    UTMIP_OP_SINK_EN = 1

    UTMIP_ON_SRC_EN = 1
10. Wait for about 10 microseconds to allow the battery charger detector to settle.
11. Now check that the VDAT_DET_CHG_DET and VDAT_DET_STS bits in the USB_PHY_VBUS_WAKEUP_ID register are both set to 1. This indicates connection to a USB compliant dedicated charger that supports up to 1.5A current. If these bits are not set, then the connection is to a PC host, where the downstream port supports a standard 500mA current when the high power device is enumerated.
12. Set the following bits in the UTMIP_BAT_CHRG_CFG0_0 register to:

    UTMIP_OP_SRC_EN = 0

UTMIP_ON_SINK_EN = 0

UTMIP_OP_SINK_EN = 0

UTMIP_ON_SRC_EN = 0

13. Now normal USB operation can continue if connected to PC host.

Primary Charger Detection

1. Set OP_SRC_EN and ON_SINK_EN.
2. Set XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[OP_SRC_EN].
3. Set XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[ON_SINK_EN].
4. Wait for about 10 microseconds.
5. Check XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[VDAT_DET] and XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[ZIN].

   D- is larger than VLGC when both VDAT_DET and ZIN are 1

   D- is between VLGC and VDAT_REF when VDAT_DET is 1 and ZIN are 0

   D- is smaller than VDAT_REF when both VDAT_DET and ZIN are 0
6. Clear OP_SRC_EN and ON_SINK_EN.
7. Clear XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[OP_SRC_EN].
8. Clear XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[ON_SINK_EN].

Secondary Charger Detection

1. Set ON_SRC_EN and OP_SINK_EN.
2. Set XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[ON_SRC_EN].
3. Set XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[OP_SINK_EN].
4. Wait for about 10 microseconds.
5. Check XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[VDAT_DET] and XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[ZIP].

   D- is larger than VLGC when both VDAT_DET and ZIP are 1

   D- is between VLGC and VDAT_REF when VDAT_DET is 1 and ZIP are 0

   D- is smaller than VDAT_REF when both VDAT_DET and ZIP are 0
6. Clear ON_SRC_EN and OP_SINK_EN.
7. Clear XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[ON_SRC_EN].
8. Clear XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0[OP_SINK_EN].
9. Write 1’ to ‘Halt EP N OUT’ field in ‘Halt Registers’

Step 3

- Poll the Nth bit in ‘State Change EP IN/OUT’ register to make sure that EP has been halted by the hardware.

Note: *Software should clear ‘State Change EP IN/OUT’ register field by writing ‘1’ to it.*

22.8.11.1 Pause an Endpoint

The endpoint could be put into pause state either by software or hardware in the Device Mode by writing to ‘Pause EP N IN’ or ‘Pause EP N OUT’ registers fields. When in paused state, the Device Mode controller will respond with NRDY TPs for any host requests targeted at this endpoint.

The sequence to pause an Nth EP is as follows,

Step 1

- If ‘Run’ bit in the control register is set
  - Then proceed to Step 2

Else

  - Exit

Step 2

- To pause Nth IN endpoint, write ‘1’ to ‘Pause EP N IN’ field in ‘Pause Register’
- To halt Nth OUT endpoint, write ‘1’ to ‘Pause EP N OUT’ field in ‘Pause Register’

Step 3

- Poll the Nth bit in ‘State Change EP IN/OUT’ register to make sure that EP has been paused by the hardware.

Note: *Software should clear ‘State Change EP IN/OUT’ register field by writing ‘1’ to it.*

Enabling ISO Endpoints

It is the software responsibility to make sure that it pause all non-ISO endpoints while enabling any ISO EP in Device Mode. This will make sure that there are resources available in the Device Mode controller while enabling any ISO endpoint. The non-ISO endpoints could be re-enabled once all ISO EPs are enabled.

Note: *If the control endpoint is ‘halted’ by the software then the device controller will re-enable the control endpoint as soon as a valid SETUP packet is received on the link. This means the hardware will reset the value in “HALT” register.*

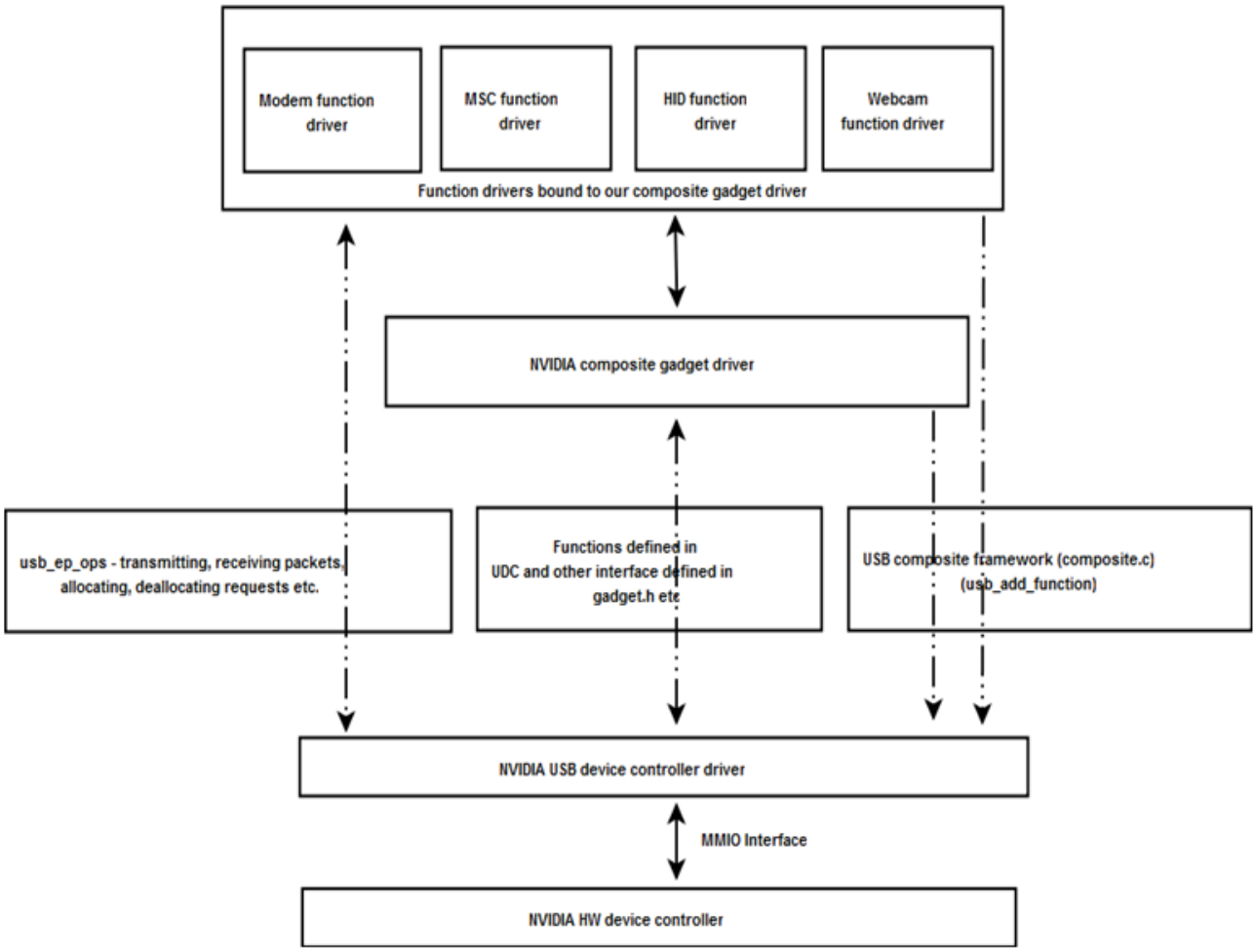
22.8.11.2 Flow Control Threshold

This is purely for the use of the device controller. The software is responsible to enable/disable this feature and to program the threshold number for the consecutive IN/OUT request being flow controller.

22.9 XUSB Programming Guidelines for Device Mode

The pictorial representation of the Device Mode software stack is shown in the below figure.

Figure 60: Device Mode Software Stack



22.9.1 Initialization

The sequence of steps to be performed to initialize the device controller is as follows,

Step 1

- Allocate and initialize all device controller related memory data structures as follows,
  - Device controller event ring segment table and the event ring segments that it points to.
  - Control endpoint contexts (one IN and one OUT) and the transfer rings that they point to.
- Initialize the event ring segment table size register with number of entries in the event ring segment table.
- Initialize the event ring segment table base address register with the physical memory address of the event ring segment table.
- Initialize the event ring segment table dequeue pointer register with the physical memory address of the event ring segment pointed by event ring segment table entry 0.

Step 2

- Initialize the endpoint context register with the physical memory address of the first entry in the array of endpoint contexts (control endpoints).

Step 3

- Enable the Device Mode controller by writing ‘1’ to enable bit (Enumeration mode) in the capability control register.

- Update the ‘Device Address’ field in the control register upon successfully receiving the ‘Set_Address’ request from the host.

Step 4

- Wait for ‘Set_Configuration’ request from the host.
  - Initialize the required endpoint contexts and the transfer rings depending upon the configuration/interface selected.
  - Software has to initialize all 32 EP contexts at this step.
  - Check for the correctness of the requested configuration with ‘Set_Configuration’ as discussion in the subsequent sections.
- Write ‘1’ to ‘Run’ field (Run mode) in the control register as the last step in completing Set_Configuration request from the host.
  - The hardware will fetch all 32 EP contexts before setting ‘Run’ bit to ‘1’. The valid EPs are identified by looking at EP.Type field in the endpoint context.

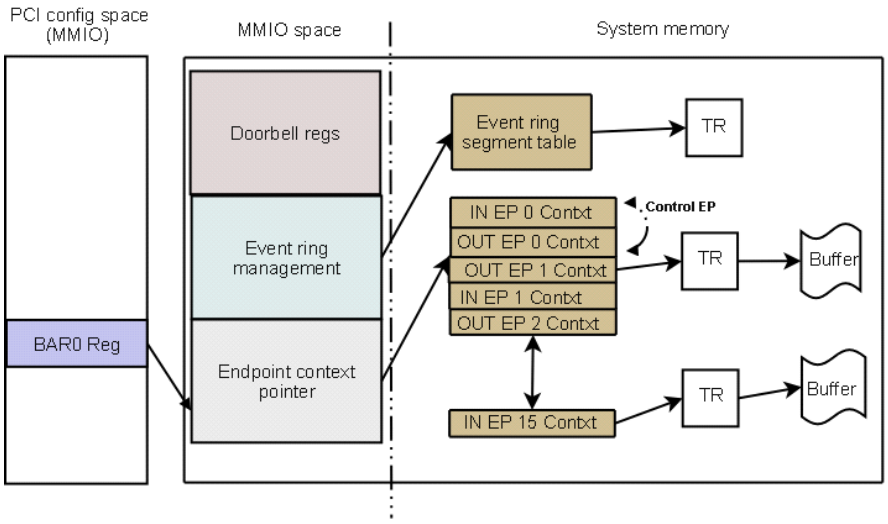
Notes:

- *All registers (PCI configuration space) are memory mapped in Tegra platform.*
- *The device controller has the option to generate an interrupt whenever an event is posted to the event ring (including SETUP).*

22.9.2 Memory Map: Device Mode

The Device Mode software will interact with the Device Mode controller through a set of transfer rings, event rings, various data structures (contexts) and the register interface based on the xHCI specification.

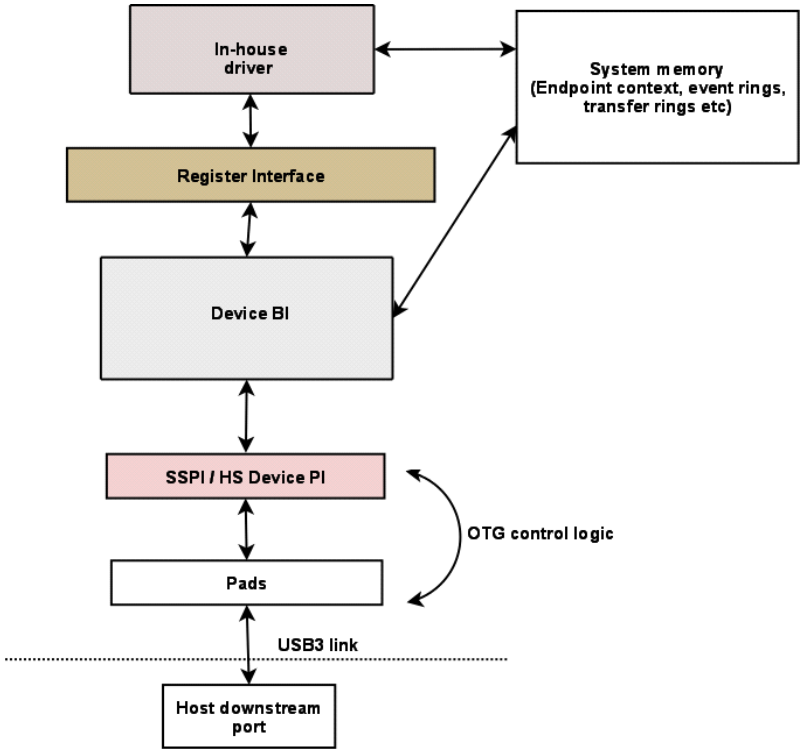
Figure 61: Memory Map for Device Mode



22.9.3 Hardware and Software Interaction

The Device Mode controller could operate either in super speed or high-speed mode.

Figure 62: Software/Hardware Interaction in Device Mode



22.9.4 Modes

22.9.4.1 Enumeration mode

The Device Mode controller will be in enumeration mode when the enable bit transitions from ‘0’ to ‘1’. While in enumeration mode, the software will services the standard USB enumeration related requests from the USB host through the default control endpoint. The standard requests need to be supported by the software are:

- Get_Descriptor
- Get_Configuration
- Get_Interface
- Get_Status
- Synch_Frame
- Set_Address
- Set_Configuration
- Set_Feature
- Clear_Feature
- Set_Interface
- Set_Isoch_Delay
- Set_Sel

The software should also support OTG specific device descriptor fields as described in Chapter 6 of the USB3 OTG specification.

The firmware needs to handle the following device requests in the debug mode:

- Get_Configuration, Get_Descriptor, and Get_Status
- Set_Feature, Clear_Feature
- Set_configuration, Set_Address
- Set_Interface, Get_Interface
  - Only one interface support. Firmware will return STALL upon receiving the requests targeting the non-default interface.
- Set_Isoch_Delay, Set_Sel, Sync_Frame
  - No need to support these requests. Firmware will return STALL upon receiving these requests.

Set_configuration

The software behavior upon receiving this request varies depending upon the device state as discussed below.

Default state: The behavior of the device is unknown upon receiving this request in default state.

Addressed state: If the specified configuration is zero, then the device remains in the addressed state. If the specified configuration value matches the configuration value from a configuration descriptor, then that configuration is selected and the device enters the configured state. Otherwise, the software should respond with STALL by writing to Nth bit in the ‘HALT’ register.

Configured state: If the specified configuration value is zero, then the device enters the address state. If the specified configuration value matches the configuration value from a configuration descriptor, then that configuration is selected and the device remains in the configured state. Otherwise, the device responds with STALL by writing to Nth bit in the ‘HALT’ register.

22.9.4.2 Steps to Handle Set_configuration

In Addressed State to set a new configuration

- Program EP Context for each endpoint that will be enabled by this Configuration
- Set Reload bit for each Endpoint being enabled
- Set Run bit after setting Reload bit for first enabled Endpoint
- Wait for all Reload bits to be cleared
- Clear all Pause bits corresponding to the Reload bits that were set
- Wait for STCHG bits to be set to indicate that Pause has been cleared
- Clear all STCHG bits
- Clear Halt bits for all configured endpoints
- Clear all STCHG bits
- Complete Status Stage for the SET_CONFIG request

In Configured State to de-configure the device (SET_CONFIG 0)

- Pause all active Endpoints
- Program EP State Field in EP context to Disabled for all Endpoints except the default control endpoint
- Clear all Augmented EP Context fields to 0 for all endpoints except for the default control endpoint
- Clear Run bit in Control register
- Wait for all STCHG bits to be set
- Clear all STCHG bits
- Complete Status stage for SET_CONFIG request

In Configured State to Select a Different Configuration

- Do all steps to deconfigure the device as mentioned in In Configured State to de-configure the device (SET_CONFIG 0).
- Do all steps to set a new configuration In Addressed State to set a new configuration.

Removing an Endpoint as part of the Select Alternate Interface

- Set Pause bit for the Endpoint to be disabled
- Wait for corresponding STCHG bit to be set
- Program EP State field in EP Context of the endpoint to Disabled
- Clear all fields in Augmented area of EP context to 0
- Set Reload bit for the Endpoint to be disabled
- Wait for STCHG bit to be set
- Clear STCHG bit

Adding an Endpoint as part of the Select Alternate Interface

- Initialize EP context for endpoint to be enabled
- Set Reload bit for the Endpoint to be enabled
- Wait for STCHG bit to be set
- Clear STCHG bit

Modifying an Endpoint as part of the Select Alternate Interface

- Do all steps in Removing an Endpoint as part of the Select Alternate Interface to remove the endpoint.
- Do all steps in Adding an Endpoint as part of the Select Alternate Interface to add the endpoint again with new parameters.

22.9.4.3 Run Mode

The Device Mode will switch from enumeration mode to run mode after receiving a “Set_Configuration” request from the host. It is the responsibility of the software to write ‘1’ to ‘Run bit’ field in control register upon successfully receiving the ‘Set_Configuration’ request from the host. The software should also move the device state from addressed state to configured state.

22.9.4.4 Test Mode

The Device Mode will switch from enumeration mode or run mode to test mode after receiving a “Set_Feature” request from the host with feature selector set to TEST_MODE. In the test mode, software should trigger the test mode by programming the ‘USB 2.0 Port Test Control’ register to match the test mode selector 1 ms after the control transfer is completed.

22.9.4.5 Device Classes Supported

The Device Mode has to support all device classes. However, the priority will be given to BOT/UAS storage classes and mobile communication classes for driver development and verification.

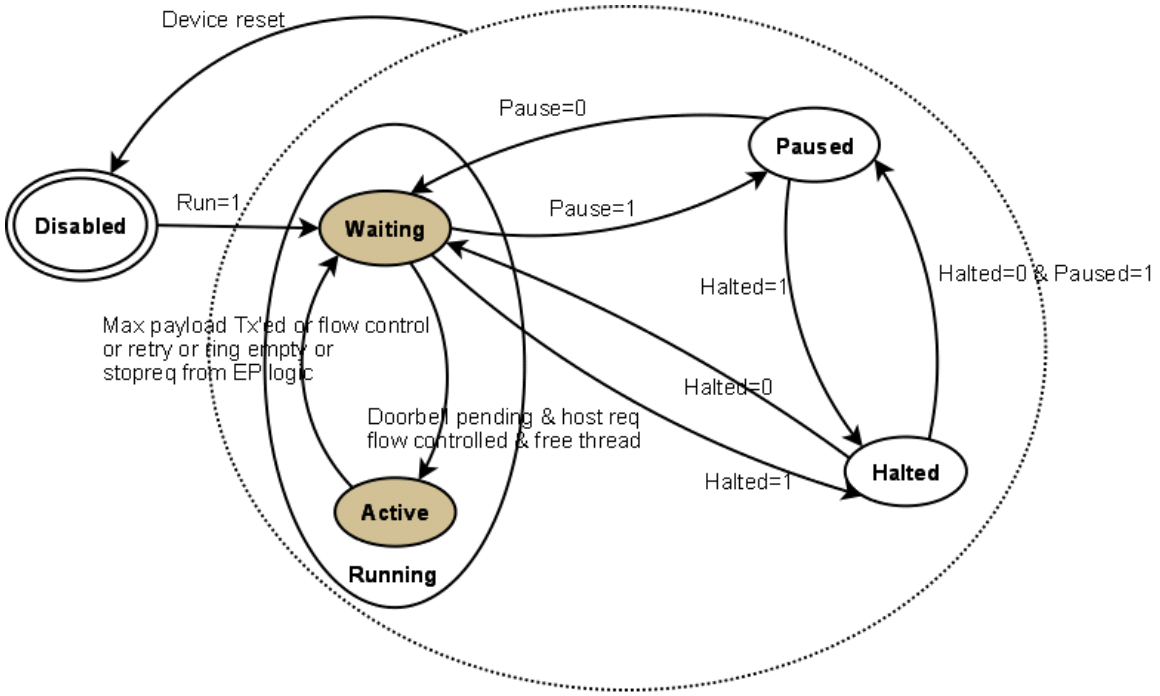
22.9.5 Endpoints

The device controller supports up to 32 endpoints (16 IN and 16 OUT). The endpoint 0 will be the control endpoint which effectively reduces the number of non-control endpoints to 15 IN and 15 OUT. Each endpoint can be set up as IN and OUT direction and as Control, Bulk, or Interrupt transfer type. Isochronous endpoint is not supported.

22.9.5.1 Endpoint Contexts

The device uses 64-byte version of the endpoint context. The data structure of the endpoint contexts for device is identical to Host Mode. However, the Endpoint context pointer points directly to Endpoint context 0 OUT (Control EP). The device controller owns the endpoint context as long as 'Run' bit is set 1.

Figure 63: EP State Transition



22.9.5.2 Default Control Endpoint

The following table lists the initial value for the control endpoint that should be setup by the software for Device Mode and firmware in debug capability.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	DW Offset
Reserved 0x0								Interval 0x0								LSA 0	MaxPStreams 0x0					Multi 0x0		Reserved 0x0					EP State 0x1			+0h
Max Packet Size 0x200																Max Burst Size 0x0								HID 0	Rsv 0	EP Type 0x4			Cerr 0x3		Rsv 0	+1h
TR Dequeue Pointer Lo																												Rsv 0x0			DCS 1	+2h
TR Dequeue Pointer HI																																+3h
Max ESIT Payload 0x0																Average TRB Length 0x8																+4h
SeqNum 0x0					SXS 0		PTD 0	Rsv 0	Event Data Transfer Length Accumulator 0x0																								+5h

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	DW Offset
Status 0x0								STD 0	RTY 0	HSP 0	CED 0	CEC 0x3		REC 0	TP 0x0		S-Byte 0x0							CProg 0x0								+6h
NumP 0x0					NumTRB 0x0					LPA 0	Rsv 0x0				Data Offset 0x0																	+7h
Scratchpad 0x0																																+8h
Scratchpad 0x0																																+9h
StopReclaimRequest 0x0								RTY 0	HSP 0	DLMM *	TLMM *	RO 1	NS 0	TC 0x0	SPing 0x0									CPing 0x0								+Ah
SlotID 0x0								RootPortNumber 0x0							Hub Address 0x0									Device Address 0x0								+Bh
DCI 0x0					Hub 0	MTT 0	LPU 0	Speed 0x0			Routing String 0x0																						+Ch
Interrupt Target 0x0										SPS 0x0	SSF 0x0				TT Port Num 0x0									TT Hub Slot ID 0x0								+Dh
EndPointLink Lo 0x0																								MRK 0	ELMM *	END 0	FRZ 0					+Eh
EndPointLink Hi 0x0																																+Fh

* M = 0 for Device Mode, indicating all data structures are located in system memory; X = 1 for debug capability, indicating all data structures are located in DDIRECT.

22.9.5.3 Halt an Endpoint

The software could put an endpoint into the halt state by programming HALT register. When in halt state, the device controller will respond with STALL for any request from the host targeted at this endpoint. The device controller will never put the endpoint into the halt state directly.

The sequence to halt an endpoint is as follows,

Step 1

- If ‘Run’ bit in the control register is set
  - Then proceed to Step 2

  Else
  - Exit

Step 2

- To halt Nth IN endpoint, write ‘1’ to ‘Halt EP N IN’ field in ‘Halt Registers’

- To halt Nth OUT endpoint, write

Step 3

- Poll the Nth bit in ‘State Change EP IN/OUT’ register to make sure that EP has been halted by the hardware.

> ***Note:*** *Software should clear ‘State Change EP IN/OUT’ register field by writing ‘1’ to it.*

22.9.5.4 Pause an Endpoint

The endpoint could be put into pause state either by software or hardware in the Device Mode by writing to ‘Pause EP N IN’ or ‘Pause EP N OUT’ registers fields. When in paused state, the Device Mode controller will respond with NRDY TPs for any host requests targeted at this endpoint.

The sequence to pause an Nth EP is as follows,

Step 1

- If ‘Run’ bit in the control register is set
  - Then proceed to Step 2

  Else
  - Exit

Step 2

- To pause Nth IN endpoint, write ‘1’ to ‘Pause EP N IN’ field in ‘Pause Register’
- To halt Nth OUT endpoint, write ‘1’ to ‘Pause EP N OUT’ field in ‘Pause Register’

Step 3

- Poll the Nth bit in ‘State Change EP IN/OUT’ register to make sure that EP has been paused by the hardware.

> ***Note:*** *Software should clear ‘State Change EP IN/OUT’ register field by writing ‘1’ to it.*

Enabling ISO Endpoints

It is the software responsibility to make sure that it pause all non-ISO endpoints while enabling any ISO EP in Device Mode. This will make sure that there are resources available in the Device Mode controller while enabling any ISO endpoint. The non-ISO endpoints could be re-enabled once all ISO EPs are enabled.

> ***Note:*** *If the control endpoint is ‘halted’ by the software then the device controller will re-enable the control endpoint as soon as a valid SETUP packet is received on the link. This means the hardware will reset the value in “HALT” register.*

22.9.5.5 Flow Control Threshold

This is purely for the use of the device controller. The software is responsible to enable/disable this feature and to program the threshold number for the consecutive IN/OUT request being flow controller.

22.9.6 Transfer Rings

The Device Mode controller makes use of transfer ring mechanism to pass the work from software to hardware. The operation/management of the transfer ring in Device Mode will be similar to Host Mode. Please refer to section 4.9 in the xHCI spec to understand the transfer ring management. The software will initialize the required number of transfer rings depending upon the configuration chosen. The device and will make use of the TRBs and TDs to pass on the work to the hardware.

22.9.6.1 Add a TRB to the Transfer Ring

The sequence of steps to add a TRB to the transfer ring will be as follows:

Step 1

- Make sure that a free entry is available to place the new TRB on to the transfer ring

Step 2

- Write the TRB to the transfer ring pointed by the enqueue pointer
  - Initialize the ‘Cycle bit’ field in the TRB to the PCS value.

Step 3

- Increment the enqueue pointer by the size of the TRB

Step 4

- Inform the hardware to fetch the TRB by writing to the doorbell register present in the MMIO space of the device controller.

22.9.6.2 Doorbell Registers

The software has to ring the doorbell upon adding a new TRB to a transfer ring. The Device Mode controller will start processing the transfer ring after the doorbell from the software.

Note: *The software should make sure that all tasks required to enable the endpoints of the selected configuration (Set_Configuration) are performed before initiating the transaction to acknowledge the status request. This is to avoid the case wherein the received host requests could be dropped due to delay in enabling the EPs at the device side.*

Control Endpoint

There is one major difference while sending a doorbell register write to the control endpoint in the Device Mode compared to Host Mode. The difference is illustrated in the below sequence to handle the SETUP request.

Step 1

- A SETUP packet event TRB is received by the software/firmware.
  - The hardware will pass an unique control sequence number (Say X) as part of the SETUP packet event TRB. Please check section 6.2.1 in this document to know the format of the SETUP packet event TRB.

Step 2

- Software/firmware prepares the status and data stage TRB for the received SETUP request from the opposite side.

Step 3

- Software/firmware informs the hardware about the data and status stage TRB by writing 00h or 01h to ‘Doorbell Target’ and writing X to ‘control sequence number’ field in the doorbell registers

22.9.7 TRB Format

The TRB format of the various endpoints will be similar to the format defined in the xHCI spec for the similar endpoints.

22.9.7.1 Bulk and Interrupt Endpoints

The bulk and interrupt endpoints will use the normal TRB defined in the xHCI specification for their data transfers. Please refer to section 6.4.1.1 in the xHCI specification.

22.9.7.2 Control Endpoints

The control endpoints will use the data and status stage TRB as defined in section 6.4.1.2.2 and 6.4.1.2.3 of the xHCI specification. The data and status stage of the control transfer will make use of EP 0 context.

22.9.7.3 Isochronous Endpoints

The isochronous endpoints use the Isoc TRB defined in section 6.4.1.3 of the xHCI spec. The debug mode does not support isochronous endpoints.

22.9.7.4 Micro-frame Synchronizer

The Device Mode controller implements the local frame timer to track the micro-frames and in turn frame ID. The software could read the current Frame ID by accessing 'FrameID' and 'uFrameID' registers in MMIO space.

22.9.8 Event Ring

The operation and management of the event ring will be similar to device debug capability mode described in xHCI spec. The Device Mode will support one event ring defined through event ring segment table, event ring segment table size, event ring segment table base address, and event ring dequeue pointer registers in the register interfaces. Here, hardware will own the event ring and the software/driver will be the consumer of the event ring.

Note: *Software should always program and use 2 segments for the event ring.*

22.9.8.1 Event Ring Non-Empty

The hardware will generate an interrupt upon posting an event TRB to the system memory. The 'Interrupt Pending' field in the status register (offset 0x24) will be set upon posting the event TRB. The interrupt will be generate depending upon whether the 'Interrupt Enable' field in the control register (offset 0x20) is set or not.

The event TRBs are generated by the hardware for these transactions,

- IN DMA engine
- Out DMA engine
- Control (Setup), Flow control threshold
- Port status change

22.9.8.2 Setup Event TRB

The device controller posts a setup event TRB to the event ring upon receiving a setup transaction from the host. The hardware will also generate an interrupt upon posting the event.

The setup data packet will be sent as part of the setup event TRB posted to the event ring. The software will make use of Endpoint 0 (IN or OUT) to send data stage and status stage transactions in response to SETUP request from the host. The device controller will report errors if the sequence number or the direction of the data/status stage TRB does not match the control transfer.

Notes:

- *If a SETUP request is received when the control EP state is set to "Halted" by software then the hardware will clear the halt condition.*
- *A "serial number" field will be included as part of the SETUP event TRB posted from the hardware. This will be useful for the hardware to identify the valid SETUP transaction.*
- *For example, a new SETUP request was received before the successful completion of the previous SETUP request of the same control endpoint.*

- *An event TRB will be posted by the hardware when there is a mismatch in the serial number stored locally (points to latest SETUP) in the hardware and the serial number present in the data/status TRBs fetched as part of the previous SETUP transaction.*

22.9.8.3 Port Status Change Event Generation (PSCEG)

The hardware places an event TRB of type “PSCEG” whenever the following fields in the control register change,

- Connect status change (CSC)
- Port reset change (PRC)
- Port link state change (PLC)
- Port configuration link error (CEC)

It is the responsibility of the software to check and clear the above register fields upon receiving the PSCEG event TRBs. The format of the PSCEG event TRB is similar to xHCI defined event TRB with port ID being set to 0x0.

22.9.8.4 Host Rejected Stream Selection Transfer Event TRB

This event TRB will be posted to indicate the host rejection for a particular StreamID selected by the device.

22.9.8.5 Prime Pipe Received Transfer Event TRB

This event TRB will be posted to indicate that host has issued a Prime Pipe transaction for a Bulk Stream Endpoint. Software can use this Event to reschedule a StreamID which has previously been rejected by host. Software should not ring Doorbell to a Stream Enabled Endpoint which had the Stream ID selection rejected by host, until it receives a Prime Pipe Received Transfer Event TRB.

22.9.8.6 Stream NumP Error Transfer Event TRB

This Event TRB will be posted to indicate that host has violated NumP and PP fields in an ACK TP sent to a Stream enabled endpoint. Software should HALT the Stream Enabled endpoint on receiving this Transfer Event TRB.

22.9.8.7 Babble Detected Error Transfer Event TRB

This Event TRB will be posted to indicate that host has sent excessive data payload that violates data length field in DP. Software should wait till hardware setting the EP State to Stopped, then clear the EP State and set the Halt bit of the endpoint to trigger the endpoint to respond subsequent host request with STALL.

22.9.8.8 Interrupts

The Device Mode controller will need to support both MSI (one vector) and legacy interrupts. The hardware will generate an interrupt upon posting an event TRB to the event ring. The interrupt generation is moderated (as in Host Mode) and there are dedicated registers to control the interrupt generation interval (‘Interrupt modulation interval’) and to track the current interrupt counter value (‘Interrupt modulation counter’).

22.9.9 Port

22.9.9.1 Generating a device notification packet

The registers to generate the device notification packet are present in the MMIO space. However, these registers are not exposed to the debug software in debug mode and are implemented as internal registers accessible by the firmware. These registers valid are only for super speed mode and when the device controller is enabled.

The software sequence to generate a device notification packet is as follows,

Step 1

- Set the ‘notification type’ bit field in the device notification register

Step 2

- Set the ‘notification type specific’ bits field in the device notification register

Step 3

- Write ‘1’ to ‘device notification packet fire’ bit to generate the packet. The bit will be cleared by the hardware once the device notification packet has been sent on the USB link.

22.9.9.2 Host Connection

The device controller will generate a “Port Status Change Event” upon detection of the host connection. It will also set the “Interrupt pending” bit in the status register whenever event ring is non-empty. An interrupt will be generated to the software if “Interrupt Enable” bit is set.

Note: *‘Link Status Event Enable’ field in the control register should be set to generate the port status change events due to change in the port link status.*

22.9.9.3 Port Reset Handling

Hardware will generate a ‘port status change’ event TRB upon detecting the port reset from the host. Upon receiving the event TRB, the software will check ‘PR’ bit in the control register for any port reset. The following sequence is executed upon port reset detection,

Step 1

- Poll ‘PRC’ field in the control register to make sure that the reset sequence is complete.
- Clear the ‘PRC’ field in the control registers once the reset sequence is completed.

Step 2

- Make sure that the port is enabled by looking at ‘Port Enabled/Disabled’ field of the port status and control register.

Step 3

- Reset the internal states of the software/driver.
- Remove any pending TDs
- Restart transfer rings management

Step 4

- The device will be in enumeration mode waiting to be configured from the host.

Step 5

- Change the device state to ‘addressed’ once the ‘Set_address’ request is received from the host.
- Write the address assigned to ‘Device Address’ field in the control register.

Step 6

- Wait for ‘Set_Configuration’ request from the host.
  - Initialize the required endpoint contexts and the transfer rings depending upon the configuration/interface selected.
- Clear ‘Run Change’ field in the control register.
- Write ‘1’ to run bit (Run mode) in the control register as the last step in completing Set_Configuration request from the host.

Note: *The 'Link Status Event Enable' field in the control registers needs to be set to generate any PSCEG event TRBs from the device controller.*

22.9.9.4 Link States

Connect

The OTG-B device will generate a "Port Status Change Event" when Vbus presence or removal is detected. This is not true in the case of OTG A-device switching from Host Mode to device (Vbus always present).

The Device Mode will also asserts the "PME" when the Vbus presence is detected, and "PME Enable" bit and "Wake on Connection" bit are set to 0x1. The wake on signaling will be controlled by PORTSC register in the case of debug mode.

Both device and debug mode will assert its PME when Vbus is removed and, "PME Enable" bit and "Wake on Disconnection" bit are set 0x1.

Link power management

The software could write to "Port Link State" field of Port Control register to manage the link states. The "Port Link State" field will reflect the software initiated link state once the hardware completes the link state transition.

The Device Mode could initiate only SS U1 and SS U2 states. The software could initiate SS U1 and U2 entry/exit by programming the Port PM registers (With SS U2 timeout value, SS U1 Enable and SS Force Link PM Accept) as required. For high speed, the Device Mode responds to L1 depending upon the value programmed into PortPM registers (High speed L1 State). The U3/L2 state transition is always accepted by the Device Mode /debug mode (only U3).

Suspend

The software should not enable suspend, resume and remote wakeup if the port is connected in high-speed and the port is acting either as a host in OTG B-Device or as a device in OTG A-Device. The resulting behavior is unknown for the above case. The software will be responsible for EP states (disabled or not) when the links are in suspend state.

Resume: Device Initiated

The sequence of steps to generate the resume signaling is described below. The sequence will remain same for both device and debug mode.

Step 1

- The software decides to send the resume signaling to the opposite side.

Step 2

- Make sure Device Remote Wake Enable bit (USB 2.0) or Function Remote Wake Enable bit (USB 3.0) is set.

Step 3

- Transition the link to U0 by writing 'U0' to 'Port Link State' field in the port status and control register.

Step 4

- Write to the device notification register to generate a 'Function_WAKE' device notification packet (for USB3.0 only).

Resume: Host Initiated

The sequence upon detecting host initiated resume signaling is as follows,

Step 1

- A port status event TRB will be posted upon detecting the resume signaling from the host.
- The 'Port Link State' field of the port control and status register will be set to 'Resume'

Step 2

- The software/debug-software should enable the link upon resume signaling by writing ‘U0’ to ‘Port Link’ State field.

22.9.10 Device Mode DVFS Support

Software should set the halt_ltssm and ltssm_state_change_pme_en bits for all SSPI before the SSPI clocks are lowered from their operation frequencies. Software should clear the halt_ltssm and ltssm_state_change_pme_en bits for all SSPI after the SSPI clocks are increased to their operation frequencies.

Software should increase the frequencies of SSPI clocks before clearing the halt_ltssm bit to allow LTSSM state transitions for the following LTSSM state transitions requests:

- Rx.Detect to Polling
- U3 to Recovery
- U2 to Recovery (note, this might not be enabled due to the shorter U2 exit time)

Software should clear the halt_ltssm bit to allow the following state transitions requested by SSPI, since they don’t require SSPI clocks to be in operating frequency, and then set the halt_ltssm bit again to prevent further LTSSM state transitions:

- Rx.Detect to SS.Disabled
- SS.Disabled to Rx.Detect
- U3 to Rx.Detect
- U2 to Rx.Detect (note, this might not be enabled due to the shorter U2 exit time)

22.9.11 On-The-Go (OTG) Support

22.9.11.1 Initial Role

The ID pin will be used to determine the role of the devices during the initialization. It is the responsibility of the software to program ‘Reverse ID’ field in ‘USB 2.0 Port XX Capability’ register field in the OTG port control register appropriately depending upon the value in the ‘ID State’ field and the ‘OTG Enable’ field in the port control register.

- There is an option to generate an interrupt whenever the value of ‘ID State’ changes.

To configure a USB port as an OTG port, software should enable the OTG function during the port configuration step in the XUSB initialization sequence:

- OTG port enable
  - PORTN_CAP = OTG_CAP
  - REVERSE_ID = NO

In OTG port operation, the ID pin is used to determine the role of the OTG port and the VBUS pin is used to determine connection status when the OTG port is identified as a peripheral port. With ID and VBUS pins of USB ports physically connecting to external components outside the Tegra SOC, software is required to update the statuses of ID and VBUS pins to XUSB host and device controllers with the VBUS_OVERRIDE and ID_OVERRIDE fields:

- Cable not connected
  - VBUS_OVERRIDE = VBUS_OFF
  - ID_OVERRODE = ID_FLOAT
- Connecting to a host port
  - VBUS_OVERRIDE = VBUS_ON
  - ID_OVERRODE = ID_FLOAT
- Connecting to a peripheral port

- VBUS_OVERRIDE = VBUS_OFF
- ID_OVERRODE = ID_GND

22.9.11.2 Role Negotiation

The super speed devices make use of LMP packets to communicate the device roles and the USB2 device use the HNP mechanism to requests the role swap. Please refer to section 6.3 and 6.4 in USB3 spec to know more.

Super Speed: Host to Device

This is applicable only when OTG-B device acting as host decides not to remain in this role. The sequence of steps to negotiate the OTG role is as follows,

Step 1

- If ‘OTG Enable’ field in the ‘USB 2.0 Port XX Capability’ register (corresponding to ‘SS Port xx Mapping’ register) register is set then

  Go to Step 2

  Else

  Exit

Step 2

- Write the new role by programming the ‘USB 2.0 Port XX Reverse ID’ field in the ‘USB 2.0 Port XX Capability’ registers to swap the roles (opposite to the current ID pin settings).

Step 3

- Send a ‘Set_feature (B3_NTF_HOST_REL)’ request to the opposite side.
- Send a device notification packet to the opposite side by writing to the device notification registers

Step 4

- Initiate the warm reset of the port by writing ‘1’ to ‘PR’ field in the port control register. The reset should be issued within Thost_Req_Susp

Step 5

- Protocol layer will exchange the LMP packets to inform the role swap request to the other end after the reset.

Super Speed: Device to Host

The sequence of steps to negotiate the OTG role is as follows,

Step 1

- If ‘OTG Enable’ field in the OTG/EH port control register is set then

  Go to Step 2

  Else

  Exit

Step 2

- Write the new role by programming the ‘Port Type’ field in the OTG/EH port control register to host.

Step 3

- Send a device notification packet to the opposite side by writing to the device notification registers.

Step 4

If there is no port reset from the opposite end within Thost_Req_Susp then continue working in Device Mode.

Step 5

Protocol layer will send the LMP packets to inform the role swap request to the other end after the reset.

Notes:

- *Both the OTG A-Device and B-Device can request to switch to be the host by sending the Host Request LMP when they are acting as the devices.*
- *Only the OTG B-Device can request to switch to be the device by sending the Device Request LMP when it is acting as the host.*
- *The OTG A device acting as a host will not send a device notification packet to act in peripheral mode.*
- *Software should warn the users before switching the roles.*
- *If the upstream port (OTG-B) does not receive the warm reset within the timeout period then the port will transition to an error state.*

High-Speed: Device to Host

The sequence of steps to switch the role from device to host is as follows,

Step 1

- If 'OTG Enable' field in the OTG/EH port control register is set then

  Go to Step 2

  Else

  Exit

Step 2

- OTG device needs to set the 'host request flag' in the OTG status information to inform the host of the role swapping.

Step 3

- Wait for Thost_Req_Poll (Table 6-6 in USB3 OTG spec) time to make sure that the host polls the request flag.

Step 4

- Software should change the 'Port Type' in the OTG/EH port control register to host.

Step 5

- Wait for the link disconnect from the host side (suspend done by hardware and disable pull-off resistor on D+ by device) within Thost_REq_Susp (Table 6-6 in the USB3 OTG specification)

Step 6

- Reconnect in the Host Mode (done by hardware)

Notes:

- *An interrupt will be generated by the device in case there is no device connection during device to device to host role switch.*
- *Step 5 and Step 6 are taken care by the hardware once setting the 'Port type' field.*

High-Speed: Host to Device

The sequence of steps to switch the role from device to host is as follows,

Step 1

- If ‘OTG Enable’ field in the OTG/EH port control register is set then

  Go to Step 2

  Else

  Exit

Step 2

- OTG host device reads the ‘host request flag’ from the device to check for role swapping request.

Step 3

- Software should change the ‘Port Type’ in the OTG/EH port control register to host

Step 4

- Suspend the link to. This is an indication to the opposite port that role swapping process is initiated.

Step 5

- Wait for an in band reset from the opposite side to get connected in Device Mode.

Notes:

- *The OTG A device acting as a host may not send a request to switch to Device Mode.*
- *An interrupt will be generated by the hardware in case there is no in band reset from the opposite side during host to device role switch.*
- *Step 4 and Step 5 are taken care by the hardware once ‘Port Type’ is implemented.*

22.9.11.3 Session Request Signaling Protocol

When the device port is connected as OTG-B device it is possible that the opposite side (host) could disable the Vbus when there is no active session on the bus. The device side software could request the host to provide Vbus as follows.

Step 1

- The user initiates the process to start a new session with the host.

Step 2

- Enable the SRP control logic by setting ‘Session Request Signaling Enable’ field in the battery port control and status register.

Step 3

- The ‘Session Request Signaling Enable’ field will be automatically cleared by the hardware once there is a valid Vbus value.

Notes:

- *There is no need to support session request detection protocol as the device acting in OTG-A Host Mode will always keep the Vbus on in Tegra X1.*
- *The device could make use of ‘Vbus State’ field in OTG/EH port status register to detect the Vbus presence.*

22.9.11.4 Attach Detection Protocol

The Vbus will always be on in Tegra X1. Hence, there is no need to support ADP.

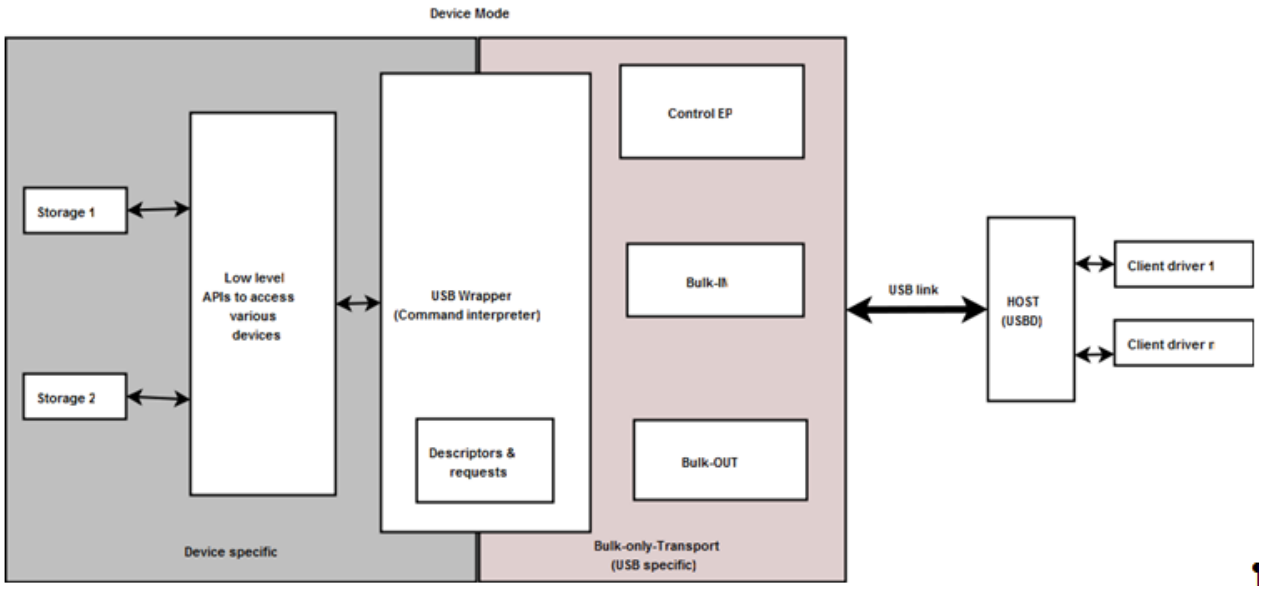
22.9.11.5 Battery Charging Sequence

The software will not be involved in the battery charging sequence. The battery charging for the dead battery provision will be taken care of by the Synopsys USB controllers.

22.9.12 Mass Storage Class

22.9.12.1 Bulk-Only Transport (BOT)

Figure 64: BOT



It makes use of one control EP, one bulk IN endpoint and one bulk OUT endpoint to communicate with the opposite side (host port). However, the throughput achieved using BOT will be less as no command queuing is supported by this protocol. In BOT, every transaction has to be completed before the next transaction could be initiated by the host. For backward compatibility, BOT interface should always be declared as first interface.

BOT requests to be supported by the device driver are listed below.

22.9.12.2 Bulk-Only Mass Storage Reset (BOMSR)

This class-specific request shall ready the device for the next command from the host. This is sent through control endpoint.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001b	11111111b	0000h	Interface	0000h	none

The sequences of actions to be taken by the driver are as follows:

Step 1

- The device driver receives the BOMSR command through the SETUP packet.

Step 2

- Decode the type of command requested.

Step 3

- Place the corresponding request on to storage controller.

Step 4

- Wait for the completion of the storage controller reset.

Step 5

- Clear the internal variable associated with all interfaces enabled for the chosen configuration. (e.g. Reset the transfer rings.)

Step 6

- Prepare and place the Data/status TRB on to the transfer ring once the storage reset is over.
  - The hardware will keep on sending the NRDY till the status TRB are placed on the transfer ring.

Notes:

- *Only BOMSR and Get Max Len requests are received through the control endpoints. All other requests are through bulk-endpoints. BOMSR is issued to clear the stall condition.*
- *The host will send CLEAR_HALT request for bulk-IN and bulk-OUT endpoints.*

22.9.12.3 Get Max LUN

The host will send this command to find out the number of logical storages supported by the device. The BOT protocol supports up to 15 LUNs (0 to 15). Most likely, the multiple LUN will not be supported with BOT.

The SETUP packet fields will look like this for the Get Max LUN.

- bmRequestType: Class, Interface, device to host
  - bRequest field set to 254 (FEh)
  - wValue field set to 0
  - wIndex field set to the interface number
  - wLength field set to 1

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100001b	11111110b	0000h	Interface	0001h	1 byte

The device will return the number of LUNs supported as part of the data stage. The device could choose to send STALL response if multiple LUNs are not supported or could send zero as part of the data stage.

The driver sequence upon receiving the ‘Get Max LUN’ will be as follows:

Step 1

- ‘Get Max LUN’ request is received as part of the SETUP request

Step 2

- If the command is not supported
  - Then HALT the endpoint by programming the ‘HATL EP xx’ register.
- Else
  - Return the number of LUNs supported as part of the data stage.

22.9.12.4 Host commands to the mass storage device

CBW – Command Block Wrapper

The host sends the CBW to the device's bulk OUT endpoint. The CBW contains a command block and other information about the command. The size of CBW is 31 bytes. The driver steps upon receiving the CBW is as follows,

Step 1

Check the validity of the CBW structure

- CBW is received after a CSW or reset.
- The CBW is 31 bytes.
- The dCBWSignature field has the correct value.

Step 2

Check that the contents are valid.

- All of the reserved bits are zero.
- The bCBWLUN field contains a supported LUN value.
- The bCBWCBLength and CBWCB fields are valid for the interface's subclass.

Step 3

Prepare to receive or send the data to host.

Step 4

If sending data then get the data from the internal storage device and place the corresponding TRBs on the transfer ring.

Step 5

If receiving the data then get the data from the host and write it to the internal storage.

Data Phase

There is no data transfer between CBW phase and CSW phase if dCBWDataTransferLength is zero.

Common Status Write (CSW)

This will be send by the device to the host once the data transfer or the command phase is completed. The size of the CSW will be 13 bytes. This will be send through the bulk-IN endpoint.

Note: *The specification does not specify the timeout for host initiated requests targeted to mass storage device. The windows operation system implements around 20-30 seconds (some source, need to check)*

22.9.12.5 STALL

The bulk-IN endpoints should be halted upon sending a STALL response to the opposite side. The STALL response will be sent in the following scenarios,

- The device sends the less data than the requested amount during bulk-IN data phase.
- If the received CBW request is invalid.

Bulk-OUT could be halted or accept and discard any received data on the endpoint (preferred).

Notes:

- *There is no support for bi-directional data transfer in a single command.*

- *BOT protocol does not support more than one outstanding CBW request to the device (command queuing)*

22.9.12.6 Standard Descriptors

The descriptor data returned by the device to the hose is speed dependent. Depending upon the current operating mode (High speed or Superspeed), the device driver will return the configuration information corresponding to that mode only.

Table 112: Device Descriptor (One per Device)

Offset	Field	Size	Value	Description
0	bLength	1	0x12	Size of this descriptor
1	bDescriptorType	1	0x1	
2	bcdUSB	2	0x0300h	
4	bDeviceClass	1	0	
5	bDeviceSubClass	1	0	
6	bDeviceProtocol	1	0	
7	bMaxPacketSize0	1	HS – 0x40 SS - 0x9	
8	idVendor	2		
10	idProduct	2		
12	bcdDevice	2		
14	iManufacturer	1		
15	iProduct	1		
16	iSerialNumber	1		
17	bNumConfigurations	1		

Table 113: Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x9	
1	bDescriptorType	1	0x2	
2	wTotalLength	2		
4	bNumInterfaces	1		
5	bConfigurationValue	1		
6	iConfiguration	1		
7	bmAttributes	1	8b’01100000	Self powered – Yes Remote wakeup -Yes
8	bMaxPower	1		Expressed in units of 8mA for super speed.

Table 114: Interface Configuration (both SS and HS)

Offset	Field	Size	Value	Description
0	bLength	1	0x9	
1	bDescriptorType	1	0x4	
2	bInterfaceNumber	1	1	
3	bAlternateSetting	1	0	UASP will be secondary setting for backward compatibility. BOT will be the primary alternate setting (0).
4	bNumEndpoints	1	2	
5	bInterfaceClass	1	0x8	
6	bInterfaceSubClass	1		

Table 114: Interface Configuration (both SS and HS)

Offset	Field	Size	Value	Description
7	bInterfaceProtocol	1	0x50	BOT
8	iInterface	1		

Table 115: Endpoint Configuration (both HS/SS)

Offset	Field	Size	Value	Description
Bulk-IN				
0	bLength	1	0x7	
1	bDescriptorType	1	0x5	
2	bEndpointAddress	1	0x81	
3	bmAttributes	1	0x2	
4	wMaxPacketSize	2	HS – 512 SS – 1024	
6	bInterval	1	00h	
Bulk-OUT				
0	bLength	1	0x7	
1	bDescriptorType	1	0x5	
2	bEndpointAddress	1	0x01	
3	bmAttributes	1	0x2	
4	wMaxPacketSize	2	HS – 512 SS - 1024	
6	bInterval	1	00h	

Notes:

- *The values of some fields change as per the device of the speed.*
- *Refer to BOT specification to know more about the descriptor details.*

Table 116: String Descriptor

Offset	Field	Size	Value	Description
0	bLength	1		
1	bDescriptor	1		
2	wString1	2		Serial number character 1
4	WString2	2		Serial number character 2
6				
8				
10	sString12	2		Serial number will be at least 12 character long

22.10 Recommended PHY Settings

Contact your local NVIDIA representative for PHY settings specific to your design.

22.11 BIAS Pad Configuration

The UTMIP BIAS pad is shared across USB_OTG and XUSB controllers and the below common bias pad settings need to be configured from the USB1 controller only.

USB1_UTMIP_BIAS_CFG0_0:

- UTMIP_BIASPD
- UTMIP_HSCHIRP_LEVEL
- UTMIP_HSSQUELCH_LEVEL
- UTMIP_HSDISCON_LEVEL_MSB
- UTMIP_HSDISCON_LEVEL
- UTMIP_ACTIVE_TERM_OFFSET
- UTMIP_ACTIVE_PULLUP_OFFSET
- UTMIP_VBUS_LEVEL_LEVEL
- UTMIP_SESS_LEVEL_LEVEL

USB1_UTMIP_BIAS_CFG1_0:

- UTMIP_FORCE_PDTRK_POWERUP
- UTMIP_FORCE_PDTRK_POWERDOWN

Following is the sequence to configure above parameters of the BIAS pad:

1. Enable the clock to the USB1 controller
2. Set any of above parameters as required.
3. Disable the clock to the USB1 controller if it is not used.

22.12 Boot ROM Initialization Sequence for USB Recovery

1. Program PLL_U.
   - Set the PLLU_BASE register fields, PLLU_DIVM, PLLU_DIVN, PLLU_VCO_FREQ, PLLU_BYPASS and PLLU_ENABLE fields as described in this document.
2. Configure USB_OTG
   a. Bring up USB_OTG clocks by writing 1 to CLK_ENB_USBD in the CLK_OUT_ENB_L register.
   b. Assert and deassert the master USBD reset in the CAR block (SWR_USBD_RST in the RST_DEVICES_L register) to bring USB_OTG out of reset.
   c. Stop the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in the UTMIP_MISC_CFG1 register to 0. This only stops the crystal clocks in the UTMIP units.
      - The default value of USB1_UTMIP_PHY_XTAL_CLOCKEN (= 1) now changes to 0.
      - To use the A Session Valid for cable detection logic, set the USB1_VBUS_SENSE_CTL field in the USB1_LEGACY_CTRL register to A_SESS_VLD (2’b11).
   d. Program the automatic PLL start times.

      Set USB1_IF_UTMIP_PLLU_ENABLE_DLY_COUNT, UTMIP_PLLU_STABLE_COUNT, UTMIP_PLL_ACTIVE_DLY_COUNT and UTMIP_XTAL_FREQ_COUNT as per the values given in this document.
   e. Program the tracking duration.

      Set USB1_IF_UTMIP_BIAS_CFG1.UTMIP_BIAS_PDTRK_COUNT as per the values given in this document.
   f. Program the debouncer length times.

      Set UTMIP_DEBOUNCE_CFG0.UTMIP_BIAS_DEBOUNCE_A field.
   g. Program various static parameters of the USB_OTG UTMIP1.

- Set UTMIP_TX_CFG0.UTMIP_FS_PREAMBLE_J to 0x1.
- Set UTMIP_BAT_CHRG_CFG0.UTMIP_PD_CHRG to 1.
- Set UTMIP_XCVR_CFG0.UTMIP_XCVR_LSBIAS_SEL to 0.
- Set the third bit of UTMIP_SPARE_CFG0 to 1. i.e., UTMIP_SPARE_CFG0[3] to 1.
- Set UTMIP_HSRX_CFG0. UTMIP_IDLE_WAIT as per the values given in this document.
- Set UTMIP_HSRX_CFG0.UTMIP_ELASTIC_LIMIT to 16.
- Set UTMIP_HSRX_CFG1.UTMIP_HS_SYNC_START_DLY to 9

h. Restart the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in the UTMIP_MISC_CFG1 register to 1 for USB_OTG.

3. Wait for cable connect on the USB_OTG UTMIP1 port. When cable is connected on USB_OTG UTMIP1 port, continue to the next step.
4. Bring UTMIP1 out of reset by writing 0 to the UTMIP_RESET bit of the USB1_IF_SUSP_CTRL register.
5. Wait until USB1_IF_USB_SUSP_CTRL.PHY_CLK_VALID is set to 1.
6. Then perform USB controller initialization.
   a. Reset the bus.
   b. Wait until the bus comes out of reset.
   c. Set the controller in Device Mode.
   d. Perform USB operations.

22.13 Performance Settings for USB Controllers

To meet USB's strict bandwidth/latency requirements, some AHB programming needs to be done. The following gives a guideline on the programming requirements to achieve maximum performance from USB:

- The burst size for the USB controller should be programmed to 8 in the USB2D_BURSTSIZE register. Both TXPBURST and RXPBURST fields should be programmed to the same value of 8.
- The ENB_FAST_REARBITRATE field for AHB_MEM gizmo should be set to 1 in the AHB_GIZMO_AHB_MEM register.
- The IMMEDIATE field for USB gizmos should be set to 1 in the AHB_GIZMO_USB, or AHB_GIZMO_USB2 register, depending on the controller in use.
- USB controllers should be set as high-priority masters on AHB by setting the bits corresponding to each USB controller to 1 in AHB_PRIORITY_SELECT field in the register AHB_ARBITRATION_PRIORITY_CTRL and setting the priority weight to 7 by setting the AHB_PRIORITY_WEIGHT field in the same register. USB master numbers are 6 for USB_OTG and 18 for USB2. The priority weight could be relaxed depending on requirements from other AHB masters in the system as required for different use cases.
- The prefetch engine needs to be set up correctly to enable prefetching of transmit data packets for USB masters. Each USB master needs one channel on the prefetch engine. There are 4 channels on the prefetch engine. If all 3 USB masters enable one channel at the same time, it would leave one more channel for another AHB master. Each prefetch channel is controlled by the AHB_AHB_MEM_PREFETCH_CFG[NO] register, where NO=1,2,3,4. To enable prefetch for a USB controller on a channel, program AHB_MST_ID_USB, or AHB_MST_ID_USB2 in the AHB_MST_ID field for the AHB_AHB_MEM_PREFETCH_CFG[NO] register. The ADDR_BNDRY field should be set to log2 (buffer size) according to the buffer size required for the corresponding USB master. The SPEC_THROTTLE field should be set to 0, and the INACTIVITY_TIMEOUT field should be set to 0x800.
- When a particular USB controller is in Host mode, the TXFIFOTHRES field in the USB2D_TXFILLTUNING register (offset 0x154) should be set to 0x10.

All this programming needs to be done before the RS bit in USB2D_USBCMD is set to RUN (1) for the corresponding USB controller.

22.14 USB Controller Handling of USB Resume Sequence

Following are the steps for how the USB controller handles a USB resume while the port is under PMC control:

1. The PMC maintains suspend mode on the bus.
2. When auto resume happens, the system starts restoring the USB controller.
3. Set the RUN bit from the USB controller to start SOFs.
4. The USB controller is brought back to suspend state.
5. Switch the USB bus from the PMC to the USB controller.
6. Wait until a resume complete notification from the USB controller. This is handled by the EHCI driver.
7. Further USB communication can start from here.

22.15 Accessing Falcon/CSB Registers

CSB registers can be accessed through XUSB PCI CFG register space. The CSB space registers can be mapped to a 512 byte aperture in the XUSB PCI CFG space between offsets 0x800 and 0xA00.

The index of the 512 byte page has to be programmed to the CSBRANGE register in the XUSB PCI Configuration space before accessing CSB registers through this register.

The steps to access a CSB register with address CSB_ADDR is as follows:

1. Write the page index into CSBRANGE register described above. The page index is calculated by dividing CSB_ADDR with the aperture size and using the quotient.

   PAGE_INDEX=CSB_ADDR/0x200

2. Determine the offset into the selected page for the register by using the remainder from the division.

   PAGE_OFFSET=CSB_ADDR%0x200

3. Use this offset into the CSB aperture in the XUSB Config space to read or write to the CSB register. For example, if we were to access offset 0x1_0010 in CSB address space.

   (PAGE_INDEX=0x80 and PAGE_OFFSET=0x10).

Program 0x80 to the CSBRANGE register and then access address 0x7009_8800 + 0x10 = 0x7009_8810.

22.16 USB Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

22.16.1 USB1 Controller Registers

Base address: USB

22.16.1.1 USB2_CONTROLLER_USB2D_ID_0

USB2D Identification Register

Offset: 0x0 | Read/Write: RO |Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One's complement version of ID. This field is set to 0xF9.

Bit	Reset	Description
7:0	X	ID: Configuration number. This field is set to 0x06

22.16.1.2 USB2_CONTROLLER_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for Host Mode. Set to 1.

22.16.1.3 USB2_CONTROLLER_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for Device Mode.

22.16.1.4 USB2_CONTROLLER_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

22.16.1.5 USB2_CONTROLLER_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

22.16.1.6 USB2_CONTROLLER_USB2D_GPTIMER0LD_0

The host/device controller drivers can measure time-related activities using these timer registers. These registers are not part of the standard EHCI controller.

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER0LD: This field is the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration.

22.16.1.7 USB2_CONTROLLER_USB2D_GPTIMER0CTRL_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit will reload the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt, and automatically reload the counter to begin again.
23:0	RO	X	GPTCNT: This field is the value of the running timer.

22.16.1.8 USB2_CONTROLLER_USB2D_GPTIMER1LD_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER1LD: This field is the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration.

22.16.1.9 USB2_CONTROLLER_USB2D_GPTIMER1CTRL_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit will reload the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt, and automatically reload the counter to begin again.
23:0	RO	X	GPTCNT: This field is the value of the running timer.

22.16.1.10 USB2_CONTROLLER_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x30.

22.16.1.11 USB2_CONTROLLER_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

22.16.1.12 USB2_CONTROLLER_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: Indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: Indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: Indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

22.16.1.13 USB2_CONTROLLER_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability - RO. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.
17	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to USBCMD HIRD field, POSTSCx SSTS and DA fields and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".
2	X	ASP: Asynchronous Schedule Park Capability. 1 = (Default) the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field is always 1.
1	X	PFL: Programmable Frame List Flag. 0 = System software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. 1 = System software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".

22.16.1.14 USB2_CONTROLLER_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

22.16.1.15 USB2_CONTROLLER_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller can operate as an EHCI compatible USB 2 0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller can operate as a USB 2.0 device. This field is set to 1.
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL, BA, and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

22.16.1.16 USB2_CONTROLLER_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

22.16.1.17 USB2_CONTROLLER_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if the UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if the UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
0	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if the NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

22.16.1.18 USB2_CONTROLLER_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx00000000010000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50 µs and each additional increment adds 75 µs. For example, the value 0001b equals 125 µs, and the value 1111b equals 1,175 µs (~1.2 ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control.Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval. 00h: Immediate (no threshold). 01h: 1 micro-frame. 02h: 2 micro-frames. 04h: 4 micro-frames. 08h: 8 micro-frames. 10h: 16 micro-frames. 20h: 32 micro-frames. 40h: 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.
14	RW	0x0	ATDTW: Add DTD Tripwire. This bit is used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software and will be cleared by hardware when a hazard exists such that adding a dTD to a primed endpoint may go unnoticed. 0 = CLEAR 1 = SET
13	RW	0x0	SUTW: Setup Tripwire. This bit is used as a semaphore when the 8 bytes of setup data read extracted by the firmware. If the setup lockout mode is off, then there exists a hazard when new setup data arrives and firmware is copying setup data from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. 0 = CLEAR 1 = SET
11	RW	0x1	ASPE: Asynchronous Schedule Park mode Enable. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation and is Read/Write capable.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL). Read Only. Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in Host Mode. Writing a one to this bit when Device Mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
4	RW	0x0	PSE: Periodic Schedule Enable. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size. (Read/Write). 000 = (Default). This field is Read/Write only if the Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes) 100 = 64 elements (256 bytes) 101 = 32 elements (128 bytes) 110 = 16 elements (64 bytes) 111 = 8 elements (32 bytes) Only the host controller uses this field.
1	RW	0x0	RST: Controller Reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller results in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit in Device Mode is not recommended. 0 = CLEAR 1 = SET

Bit	R/W	Reset	Description
0	RW	0x0	RS: Run/Stop: Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HCHalted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event. 0 = STOP 1 = RUN

22.16.1.19 USB2_CONTROLLER_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule. 0= Disable Asynchronous Schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE then: 1 = Periodic Schedule is enabled or 0 = Periodic Schedule is disabled. Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int).The alt_int itself is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT

Bit	R/W	Reset	Description
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1 ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1 ms during the prelude to the connect and chirp. In Host Mode, this bit will be set every 125 µs and can be used by the host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits, respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

22.16.1.20 USB2_CONTROLLER_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b0000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if the ULPI_ALT_INT bit in the USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = The host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

22.16.1.21 USB2_CONTROLLER_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in Host Mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In Device Mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current micro-frame.

22.16.1.22 USB2_CONTROLLER_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2 ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2 ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. The HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

22.16.1.23 USB2_CONTROLLER_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QHs). Only used by the device controller.

Bit	Reset	Description
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

22.16.1.24 USB2_CONTROLLER_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bx0000000xxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
30:24	RW	0x0	TTHA: Internal TT Hub Address representation. This field is used to match the Hub Address field in Queue Head (QH) and split isochronous transaction descriptor (siTD) to determine if the packet is routed to the internal transaction translator (TT) for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in QH/siTD.
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set). This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	RO	X	TTAS: Embedded TT Async Buffers Status. (Read Only). This read only bit will be 1 if one or more transactions are being held in the embedded TT Async. Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

22.16.1.25 USB2_CONTROLLER_USB2D_BURSTSIZE_0

USB2D Burst Size Register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

22.16.1.26 USB2_CONTROLLER_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in Host Mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode

22.16.1.27 USB2_CONTROLLER_USB2D_ULPI_VIEWPORT_0

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Notes:

- *Writes to the ULPI through the viewport can substantially harm standard USB operations. Currently no usage model has been defined where software should need to execute writes directly to the ULPI PHY. See exception regarding optional features below.*
- *Executing read operations through the ULPI Viewport should have no harmful side effects to standard USB operations.*

There are two operations that can be performed with the ULPI Viewport: wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32-bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

USB2D ULPI Viewport Register

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET

Bit	R/W	Reset	Description
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

22.16.1.28 USB2_CONTROLLER_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address - RW. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in Host Mode. If in Device Mode it will be read only and always equal to 0000000b.
24:23	RO	X	SSTS: Suspend Status - RO. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in Host Mode. If in Device Mode, it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED 3 = PERIPH_NORESP_ERR
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in Device Mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in Device Mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in Device Mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in Device Mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in Device Mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value Specific Test. 0000b: Not enabled. 0001b: J_ STATE. 0010b: K_STATE. 0011b: SEQ_NAK. 0100b: Packet. 0101b: FORCE_ENABLE. 0110b to 1111b: Reserved. | Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always be 0.
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits is: 00b = SE0. 01b = K-state. 10b = L-state. 11b = Undefined. The value of this field is undefined if Port Power (PP) is zero in Host Mode. In Host Mode, the use of line-state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In Device Mode, the use of line-state by the device controller driver is not necessary. 0 = SE0 1 = K_STATE 2 = J_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified, the Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When this bit is set to zero, the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD[27:26]. This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET

Bit	R/W	Reset	Description
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/ Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend. When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero), the results are undefined. This field is zero if Port Power (PP) is zero in Host Mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/ driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in Host Mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero.) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default). In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in Host Mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED

Bit	R/W	Reset	Description
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/ device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in Host Mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status. In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in Host Mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

22.16.1.29 USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available in Device Mode. This register shares the same address with the HOSTPC1 register (which is used only in Host Mode).

USB2D Host Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available in Host Mode. This register shares the same address with the DEVLC register (which is used only in Device Mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER 4 = HSIC
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled, and every time the port enters the disconnect state it will also enter low power state,disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled (in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register (Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the register USBMODE. 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT

Bit	R/W	Reset	Description
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Write a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In Device Mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In Host Mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set - RW. Default = 00b. This bit field is **valid during Host Mode Only**. For Device Mode, this bit is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power - RW. Default = 0b. This bit field is **valid during Device Mode Only**. In Host Mode, it is part of the ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token - RW. Default = 0000b. This bit field is **valid during Host Mode Only**. For Device Mode, bits [19:18] are reserved and bit 17 is ASUS, while bit 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.
16	RW	0x0	STL: STALL reply to LPM token - RW. Default = 0b. This bit field is **valid during Device Mode Only**. In Host Mode, it is part of the ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT). 0 = DISABLE 1 = ENABLE
15:12	RW	0x0	LPMFRM: Auto LPM SOF Threshold - RW. Default = 0000b. This bit field is **valid during Host Mode Only**. For Device Mode, this field is reserved. This holds the SOF counter threshold. When the number of SOFs with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: bmAttributes - RO. Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.
0	RW	0x0	NYT_ASUS: NYET reply to LPM token - RW. Default = 0b. This bit is NYT during Device Mode. Details follow: When this bit is to '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. Details follow: This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

22.16.1.30 USB2_CONTROLLER_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx100x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0). PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET

Bit	R/W	Reset	Description
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x1	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: OTG Termination. This bit must be set when the OTG device is in Device Mode, this controls the pulldown on DM. 0 = NO_OTG_TERM 1 = OTG_TERM
1	RW	0x0	VC: VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP. 0 = NO_VBUS_CHRG 1 = VBUS_CHRG
0	RW	0x0	VD: VBUS_Discharge. Read/write. Setting this bit causes Vbus to discharge through a resistor. 0 = NO_VBUS_DISCHRG 1 = VBUS_DISCHRG

22.16.1.31 USB2_CONTROLLER_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay, Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in Host Mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. No functionality is implemented for this, so software should not use this bit.
4	RO	X	SDIS: Stream disable: 1: Streaming is disabled - helpful to avoid overruns/underruns when the system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In Device Mode, this bit controls the behavior of the setup lockout mechanism. 0: Setup lockout is ON (default). 1: Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this bit should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED

Bit	R/W	Reset	Description
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default]. 01 = Reserved. 10 = Device Controller. 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

22.16.1.32 USB2_CONTROLLER_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

22.16.1.33 USB2_CONTROLLER_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

22.16.1.34 USB2_CONTROLLER_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD

22.16.1.35 USB2_CONTROLLER_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

22.16.1.36 USB2_CONTROLLER_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH

22.16.1.37 USB2_CONTROLLER_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

22.16.1.38 USB2_CONTROLLER_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

22.16.1.39 USB2_CONTROLLER_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

22.16.1.40 USB2_CONTROLLER_USB2D_ENDPTCTRLn_0

USB2D Endpoint Control 1 through 15 registers have the same field definitions and reset value. In the register offset, the value n is the register number (1 through 15).

USB2D Endpoint Control n Register

Offset: 0x220 + (n – 1) * 0x04 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

22.16.2 USB1 Controller Interface Registers

These are used to generate the actual RTL registers for USB1 controller interface.

Base Address: USB

22.16.2.1 USB1_IF_USB_SUSP_CTRL_0

This register controls the suspend and resume behavior of USB controller/PHY.

USB Suspend Control Register

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00xxxxxx000xxx01000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1 ms of spec. Used for Host mode ONLY. 0 = DISABLE 1 = ENABLE
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY. Enabling this will only cut off clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running. 0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter. USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode. Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on USB. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to USB PHY. 0 = Active low (default). 1 = Active high. This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever the USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear. Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (Device Mode). When enabled (1), the USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in Device Mode, it is not applicable when USB controller is in Host Mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (Device Mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in Device Mode, it is not applicable when USB controller is in Host Mode. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB Host Modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable. If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status. This bit is set whenever USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

22.16.2.2 USB1_IF_USB_PHY_VBUS_SENSORS_0

This register controls the OTG VBUS sensors in the USB PHY. There are 4 VBUS sensors:

- A_VBUS_VLD
- A_SESS_VLD
- B_SESS_VLD
- B_SESS_END

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/BPMP-Lite by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: *Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.*

USB PHY VBUS SENSORS Control Register

Offset: 0x404 | Read/Write: R/W | Reset: 0bx0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B

Bit	R/W	Reset	Description
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable. Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status. This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable. Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status. This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable. Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status. This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	B_SESS_END_SW_VALUE: B_SESS_END software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This is only valid when B_SESS_END_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	B_SESS_END_SW_EN: B_SESS_END software enable. Enable Software Controlled B_SESS_END Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status. This is set to 1 whenever B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	B_SESS_END_CHG_DET: B_SESS_END change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_END. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	B_SESS_END_INT_EN: B_SESS_END interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_END_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

22.16.2.3 USB1_IF_USB_PHY_VBUS_WAKEUP_ID_0

This register controls the battery charger (VDCD_DET, VDAT_DET), VBUS_WAKEUP and ID sensors. The following sensors are in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/BPMP-Lite by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: *Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.*

There are debouncers for VDAT_DET and VDCD_DET. These use separate debouncers -CHRG_DEBOUNCE_PERIOD_A and CHRG_DEBOUNCE_PERIOD_B. The debounce values for them are controlled by the register UTMIP_CHRG_DEB_CFG0, fields UTMIP_CHRG_DEBOUNCE_PERIOD_A and UTMIP_CHRG_DEBOUNCE_PERIOD_B. For each sensor, we can select whether to use CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: *Do not set either UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.*

USB PHY VBUS Wakeup and ID Control Register

Offset: 0x408 | Read/Write: R/W | Reset: 0b00000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
31	RW	0x0	DIV_DET_EN: Battery charger divider detection enable. This goes to the USB2OTG pad. 0 = DISABLE 1 = ENABLE
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
29	RW	0x0	VDCD_DET_DEB_SEL_B: VCDT_DET debounce A/B select. Selects the debounce value from UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_CHRG_DEB_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	VDCD_DET_SW_VALUE: VDCD_DET software value. Software should write the appropriate value (1/0) to set/unset the VDCD_DET status. This is only valid when VDCD_DET_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	VDCD_DET_SW_EN: VDCD_DET software enable. Enable Software Controlled VDCD_DET. Software sets this bit to drive the value in VDCD_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
26	RO	X	VDCD_DET_STS: VDCD_DET status. This is set to 1 whenever VDCD_DET sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	VDCD_DET_CHG_DET: VDCD_DET change detect. This field is set by hardware whenever a change is detected in the value of VDCD_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	VDCD_DET_INT_EN: VDCD_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDCD_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
23	RO	X	VOP_DIV2P7_DET: This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
22	RO	X	VOP_DIV2P0_DET:This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET debounce A/B select. Selects between the two debounce values UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	VDAT_DET_SW_VALUE: VDAT_DET software value. Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: VDAT_DET software enable. Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status. This is set to 1 whenever the VDAT_DET sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	VDAT_DET_CHG_DET: VDAT_DET change detect. This field is set by hardware whenever a change is detected in the value of VDAT_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	VDAT_DET_INT_EN: VDAT_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDAT_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
15	RO	X	VON_DIV2P7_DET: This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET
14	RO	X	VON_DIV2P0_DET: This read-only status bit is from the battery charging divider circuit of the USB2OTG pad 0 = UNSET 1 = SET
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS wakeup software value. Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	VBUS_WAKEUP_SW_EN: VBUS wakeup software enable. Enable Software Controlled VBUS_WAKEUP. Software sets this bit to drive the value in VBUS_WAKEUP_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	VBUS_WAKEUP_STS: VBUS wakeup status. This is set to 1 whenever VBUS_WAKEUP sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	VBUS_WAKEUP_CHG_DET: VBUS wakeup change detect. This field is set by hardware whenever a change is detected in the value of VBUS_WAKEUP. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	VBUS_WAKEUP_INT_EN: VBUS wakeup interrupt enable. If this field is set to 1, an interrupt is generated whenever VBUS_WAKEUP_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
7	RO	X	STATIC_GPI: Static GPI status 0 = UNSET 1 = SET
6	RW	0x1	ID_PU: ID pullup enable. Set to 1. 0 = DISABLE 1 = ENABLE
5	RW	0x0	ID_DEB_SEL_B: ID debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	ID_SW_VALUE: ID software value. Software should write the appropriate value (1/0) to set/unset the ID status. This is only valid when ID_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	ID_SW_EN: ID software enable. Enable Software Controlled ID. Software sets this bit to drive the value in ID_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
2	RO	X	ID_STS: ID status. This is set to 1 whenever ID sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	ID_CHG_DET: ID change detect. This field is set by hardware whenever a change is detected in the value of ID. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	ID_INT_EN: ID interrupt enable. If this field is set to 1, an interrupt is generated whenever ID_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

22.16.2.4 USB1_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS/ID Status Register

Offset: 0x40c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
14	X	ID_DIG_C_ALT: IDDIG_C alternate status 0 = UNSET 1 = SET
13	X	ID_DIG_C: IDDIG_C status 0 = UNSET 1 = SET
12	X	ID_DIG_B_ALT: IDDIG_B alternate status 0 = UNSET 1 = SET
11	X	ID_DIG_B: IDDIG_B status 0 = UNSET 1 = SET
10	X	ID_DIG_A_ALT: IDDIG_A alternate status 0 = UNSET 1 = SET
9	X	ID_DIG_A: IDDIG_A status 0 = UNSET 1 = SET
8	X	VDCD_DET_ALT: VDCD_DET alternate status 0 = UNSET 1 = SET

Bit	Reset	Description
7	X	VDAT_DET_ALT: VDAT_DET alternate status 0 = UNSET 1 = SET
6	X	A_SESS_VLD_ALT: A_SESS_VLD alternate status 0 = UNSET 1 = SET
5	X	B_SESS_VLD_ALT: B_SESS_VLD alternate status 0 = UNSET 1 = SET
4	X	ID_DIG_ALT: ID alternate status 0 = UNSET 1 = SET
3	X	B_SESS_END_ALT: B_SESS_END alternate status 0 = UNSET 1 = SET
2	X	STATIC_GPI_ALT: Static GPI alternate status 0 = UNSET 1 = SET
1	X	A_VBUS_VLD_ALT: A_VBUS_VLD alternate status 0 = UNSET 1 = SET
0	X	VBUS_WAKEUP_ALT: Vbus wakeup alternate status 0 = UNSET 1 = SET

22.16.2.5 USB1_IF_USB_INTER_PKT_DELAY_CTRL_0

This controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to Host Mode as device never transmits two packets in a row.

Inter Packet Delay Control

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. Software should not change this.

22.16.2.6 USB1_IF_USB_RSM_DLY_0

USB Controller Resume Signaling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in Host Mode.

22.16.2.7 USB1_IF_SPARE_0

Note: *Software neither reads nor writes this register.*

For ICUSB PADCTLs. Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix.

Bit	Reset	Description
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en, Enable HS_RESUME EOP fix. SPARE_LO[5] - usb_port_suspend_fix_en.

22.16.2.8 USB1_IF_USB1_NEW_CONTROL_0

USB Coherency and Memory Alignment Controls

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 style (0) and Tegra X1 style (1) DMA request generation mechanism. 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE.

22.16.3 USB1 UTMIP Configuration Registers

Note: *Current UTMIP configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers*

Base Address: USB

22.16.3.1 USB1_UTMIP_XCVR_CFG0_0

UTMIP Transceiver Cell Configuration Register 0

Offset: 0x808 | Read/Write: R/W | Reset: 0x20a56500 (0b00100000101001010110010100000000

Bit	Reset	Description
31:25	0x10	UTMIP_XCVR_HSSLEW_MSB: Most significant bits of HS_SLEW.
24:22	0x2	UTMIP_XCVR_SETUP_MSB: Most significant bits of SETUP.
21	0x1	UTMIP_XCVR_LSBIAS_SEL: Low speed bias selection method for USB transceiver pad
20	0x0	UTMIP_XCVR_DISCON_METHOD: Disconnect method on the USB transceiver pad
19	0x0	UTMIP_FORCE_PDZI_POWERUP: Force PDZI input into power up.
18	0x1	UTMIP_FORCE_PDZI_POWERDOWN: Force PDZI input into power down. (Overrides FORCE_PDZI_POWERUP.)
17	0x0	UTMIP_FORCE_PD2_POWERUP: Force PD2 input into power up.
16	0x1	UTMIP_FORCE_PD2_POWERDOWN: Force PD2 input into power down. (Overrides FORCE_PD2_POWERUP.)
15	0x0	UTMIP_FORCE_PD_POWERUP: Force PD input into power up.
14	0x1	UTMIP_FORCE_PD_POWERDOWN: Force PD input into power down. (Overrides FORCE_PD_POWERUP.)
13	0x1	UTMIP_XCVR_TERMEN: Enable HS termination.
12	0x0	UTMIP_XCVR_HSLOOPBACK: Internal loopback inside XCVR cell. Used for IOBIST.
11:10	0x1	UTMIP_XCVR_LSFSLEW: LS falling slew rate control.
9:8	0x1	UTMIP_XCVR_LSRSLEW: LS rising slew rate control.
7:6	0x0	UTMIP_XCVR_FSSLEW: FS slew rate control.
5:4	0x0	UTMIP_XCVR_HSSLEW: HS slew rate control. The two LSBs.

Bit	Reset	Description
3:0	0x0	UTMIP_XCVR_SETUP: SETUP[3:0] input of XCVR cell. HS driver output control. 4 LSBs.

22.16.3.2 USB1_UTMIP_BIAS_CFG0_0

UTMIP Bias Cell Configuration Register 0

Offset: 0x80c | Read/Write: R/W | Reset: 0x00c00c08 (0bx00000001100000000000110000001000

Bit	Reset	Description
30	0x0	UTMIP_IDDIG_C_VAL: See IDDIG_C_SEL.
29	0x0	UTMIP_IDDIG_C_SEL: 0: IdDig_c = IdDig_c. 1: IdDig_c = IDDIG_C_VAL.
28	0x0	UTMIP_IDDIG_B_VAL: See IDDIG_B_SEL.
27	0x0	UTMIP_IDDIG_B_SEL: 0: IdDig_b = IdDig_b. 1: IdDig_b = IDDIG_B_VAL.
26	0x0	UTMIP_IDDIG_A_VAL: See IDDIG_A_SEL.
25	0x0	UTMIP_IDDIG_A_SEL: 0: IdDig_a = IdDig_a. 1: IdDig_a = IDDIG_A_VAL.
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x1	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x1	UTMIP_IDPD_SEL: 0: Reserved. Refer to the PMC registers for this feature.
21	0x0	UTMIP_IDDIG_VAL: See IDDIG_SEL.
20	0x0	UTMIP_IDDIG_SEL: 0: IdDig = IdDig. 1: IdDig = IDDIG_VAL.
19	0x0	UTMIP_GPI_VAL: See GPI_SEL.
18	0x0	UTMIP_GPI_SEL: 0: StaticGpi = IdDig. 1: StaticGpi = GPI_VAL.
17:15	0x0	UTMIP_ACTIVE_TERM_OFFSET: Active termination control offset.
14:12	0x0	UTMIP_ACTIVE_PULLUP_OFFSET: Active 1.5K pullup control offset.
11	0x1	UTMIP_OTGPD: Power down OTG circuit.
10	0x1	UTMIP_BIASPD: Power down bias circuit.
9:8	0x0	UTMIP_VBUS_LEVEL_LEVEL: Vbus detector level.
7:6	0x0	UTMIP_SESS_LEVEL_LEVEL: SessionEnd detector level.
5:4	0x0	UTMIP_HSCHIRP_LEVEL: HS chirp detector level.
3:2	0x2	UTMIP_HSDISCON_LEVEL: HS disconnect detector level.
1:0	0x0	UTMIP_HSSQUELCH_LEVEL: HS squelch detector level.

22.16.3.3 USB1_UTMIP_HSRX_CFG0_0

UTMIP High Speed Receive Config 0

Offset: 0x810 | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of idle cycles to declare IDLE.
14:10	0xd	UTMIP_ELASTIC_LIMIT: Depth of elastic input store
9	0x0	UTMIP_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
8	0x0	UTMIP_ELASTIC_UNDERRUN_DISABLE: Do not declare underrun errors
7	0x0	UTMIP_PASS_CHIRP: When in Chirp Mode, allow chirp RX data through
6	0x0	UTMIP_PASS_FEEDBACK: Pass through the feedback, do not block it.

Bit	Reset	Description
5:4	0x0	UTMIP_PCOUNT_INERTIA: Retime the path.
3:2	0x0	UTMIP_PHASE_ADJUST: Based on incoming edges and current sampling position, adjust phase
1	0x0	UTMIP_THREE_SYNCBITS: Sync pattern detection needs 3 consecutive samples instead of 4
0	0x0	UTMIP_USE4SYNC_TRAN: Require 4 sync pattern transitions (01) instead of 3

22.16.3.4 USB1_UTMIP_HSRX_CFG1_0

UTMIP High Speed Receive Config 1

Offset: 0x814 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010011

Bit	Reset	Description
5:1	0x9	UTMIP_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UTMIP_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

22.16.3.5 USB1_UTMIP_FSLSRX_CFG0_0

UTMIP Full and Low Speed Receive Config 0

Offset: 0x818 | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, do not allow dribble
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV
28:26	0x7	UTMIP_FSLS_UPR_DRIBBLE_SIZE: Do not allow <= dribble bits
25:23	0x2	UTMIP_FSLS_LWR_DRIBBLE_SIZE: Do not allow >= dribble bits
22	0x1	UTMIP_FSLS_EOP_ENDS_AT_SE0: Only look for transitioning out of EOP
21:16	0x14	UTMIP_FSLS_KCOUNT_MAX: Number of K bits in question
15	0x1	UTMIP_FSLS_KCOUNT_LIMIT: Limit the number of bit times a K can last
14	0x0	UTMIP_FSLS_ACTIVE_ON_FULL_SYNC: Require a full sync pattern to declare the data received
13:8	0x4	UTMIP_FSLS_IDLE_WAIT_MAX: 4 bits of SEO should exceed the time limit
7	0x0	UTMIP_FSLS_IDLE_WAIT_LIMIT: Enable the reset of the state machine on extended SE0
6:1	0x14	UTMIP_FSLS_IDLE_COUNT_MAX: 20 bits of idle should end the packet if FsLsIdleCountLimitCfg=1.
0	0x1	UTMIP_FSLS_IDLE_COUNT_LIMIT: Give up on packet if a long sequence of J

22.16.3.6 USB1_UTMIP_FSLSRX_CFG1_0

UTMIP Full and Low Speed Receive Config 1

Offset: 0x81c | Read/Write: R/W | Reset: 0bxxxxx01000100110011110100000000000

Bit	Reset	Description
26	0x0	UTMIP_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
25:23	0x4	UTMIP_LS_BOUNCE_LENGTH: Number of clock cycle of LS stable
22:17	0x13	UTMIP_LS_EXTRACTION_COUNT: Phase count on which LS bits are extracted
16:11	0xe	UTMIP_LS_EOP_START_COUNT: Number of SEO clock cycles to block bit extraction
10:5	0x20	UTMIP_LS_SE0_COUNT: Only for this number of 60MHz of SEO and Idle to end packet
4	0x0	UTMIP_LS_LENIENT_DRIBBLE: Allow for large dribble in low speed mode
3	0x0	UTMIP_FS_LENIENT_DRIBBLE: Allow for large dribble in full speed mode
2	0x0	UTMIP_FS_WEAK_SYNC: Only look for a KK pattern, instead of KJKK
1	0x0	UTMIP_FS_DEBOUNCE: Whether full speed uses debouncing
0	0x0	UTMIP_FS_EOP_LENGTH: Whether full speed EOP is determined within 3(0) or 4(1) 60 MHz cycles

22.16.3.7 USB1_UTMIP_TX_CFG0_0

UTMIP Transmit Config Signals

Offset: 0x820 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: Output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects LineState change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not send SYNC or EOP

22.16.3.8 USB1_UTMIP_MISC_CFG0_0

UTMIP Miscellaneous Configurations

Offset: 0x824 | Read/Write: R/W | Reset: 0bx00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value.
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Do not block changes for 4 ms when going from LS to FS (should not happen)
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pullup inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pullup inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pulldown inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pulldown inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pullup active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pullup active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pulldown active.

Bit	Reset	Description
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pulldown active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Do not use free running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free-running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free-running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

22.16.3.9 USB1_UTMIP_MISC_CFG1_0

UTMIP Miscellaneous Configurations

Offset: 0x828 | Read/Write: R/W | Reset: 0bx10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass LineState reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use negative edge sync for line state
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: Reserved. Config moved to the Clock and Reset space.
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: Reserved. Moved to the Clock and Reset space.
5	0x1	UTMIP_RX_ERROR_CNT_CLR
4	0x0	UTMIP_RX_ERROR_CNT_EN
3	0x0	UTMIP_FLIP_FSLS_POLARITY
2	0x1	UTMIP_SUSPEND_TERMSEL
1:0	0x0	UTMIP_XCVRSEL3: Bit 0: 0: 0xa5 -> treat as KeepAlive 1: treat as regular packet Bit 1: 0: Turn on FS EOP detection 1: Turn off FS EOP detection

22.16.3.10 USB1_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid Debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals has its own debouncer, and for each of those one out of 2 debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2 MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A:

BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x82c | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

22.16.3.11 USB1_UTMIP_BAT_CHRG_CFG0_0

UTMIP Battery Charger Configuration

Offset: 0x830 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx000000xx000001

Bit	Reset	Description
13:8	0x0	UTMIP_CHRG_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor-1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
5	0x0	UTMIP_OP_I_SRC_EN
4	0x0	UTMIP_ON_SRC_EN
3	0x0	UTMIP_OP_SRC_EN
2	0x0	UTMIP_ON_SINK_EN
1	0x0	UTMIP_OP_SINK_EN
0	0x1	UTMIP_PD_CHRG: Power down charger circuit

22.16.3.12 USB1_UTMIP_SPARE_CFG0_0

UTMIP Spare Configuration Bits

Offset: 0x834 | Read/Write: R/W | Reset: 0b11111111111111110000000111111000

Bit	Reset	Description
31:0	-65032	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 3: FUSE_SETUP_SEL. Select between regular CFG value and JTAG values for UX_SETUP 4: FUSE_TERM_RANGE_ADJ_SEL. Select between regular CFG value and JTAG values for UX_TERM_RANGE_ADJ 5: FUSE_SPARE. Select between regular CFG value and JTAG values. For any ECOs. 6: FUSE_HS_SQUELCH_LEVEL. Select between regular CFG value and JTAG values 7: FUSE_HS_IREF_CAP_CFG. Select between regular CFG value and JTAG values 8: FUSE_RPD_CTRL. Select between regular CFG value and JTAG values 9: PD2_OLD_SCHEME_SEL. Select between old and new scheme. 31 to 10: Reserved

22.16.3.13 USB1_UTMIP_XCVR_CFG1_0

UTMIP Transceiver Cell Configuration Register 1

Offset: 0x838 | Read/Write: R/W | Reset: 0bxxxx00000010000010000010000010101

Bit	Reset	Description
27:26	0x0	UTMIP_XCVR_RPU_RANGE_ADJ: 1.5k pull-up resistor range shift.
25:24	0x0	UTMIP_XCVR_HS_IREF_CAP: High-speed Iref cap control for bias current stability.
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for USB transceiver pad.
21:18	0x8	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjustment on terminations.
17	0x0	UTMIP_RCTRL_SW_SET: Use a software override on RCTRL instead of automatic bias control.
16:12	0x8	UTMIP_RCTRL_SW_VAL: Encoded value to use on RCTRL when software override is enabled, 0 to 16 only.
11	0x0	UTMIP_TCTRL_SW_SET: Use a software override on TCTRL instead of automatic bias control
10:6	0x8	UTMIP_TCTRL_SW_VAL: Encoded value to use on TCTRL when software override is enabled, 0 to 16 only.
5	0x0	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x1	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input into power down. (Overrides FORCE_PDDR_POWERUP.)
3	0x0	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.

Bit	Reset	Description
2	0x1	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x0	UTMIP_FORCE_PDDISC_POWERUP: Force PDDISC input into power up.
0	0x1	UTMIP_FORCE_PDDISC_POWERDOWN: Force PDDISC input into power down. (Overrides FORCE_PDDISC_POWERUP.)

22.16.3.14 USB1_UTMIP_BIAS_CFG1_0

UTMIP Bias Cell Configuration Register 1

Offset: 0x83c | Read/Write: R/W | Reset: 0bxxxxxxxx00000000000000000101101

Bit	Reset	Description
23	0x0	UTMIP_BIAS_TRK_DONE: TRK cycle done status. UTMIP REGISTER: UTMIP_BIAS_STS0
22	0x0	UTMIP_BIAS_TRK_START_OVERRIDE: The Track cycle starts after removing power downs on TRK circuit(PD_TRK).Alternatively, this track cycle can be started with this override bit, auto-cleared by hardware after TRK_START happens.
21:14	0x0	UTMIP_BIAS_TRK_START_COUNT: The lag between PD_TRK and TRK_START.
13:8	0x0	UTMIP_BIAS_DEBOUNCE_TIMESCALE: Debouncer time scaling. Slows down debouncing by a factor-1. So 0 is 1, 1 is 2, etc.
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20 µs. For a crystal clock of 13 MHz, it should be set to 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Reserved. See the PMC registers for this functionality.
1	0x0	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x1	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

22.16.3.15 USB1_UTMIP_BIAS_STS0_0

UTMIP Bias Cell Status Register 0

Offset: 0x840 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

22.16.3.16 USB1_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det Debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals has its own debouncer and for each of those, 1 out of 2 debouncing times can be chosen (CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B).

The values of DEBOUNCE_PERIOD_A and DEBOUNCE_PERIOD_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2 MHz) / 4

So to program a 1 ms debounce for CHG_DEBOUNCE_PERIOD:

CHG_DEBOUNCE_PERIOD[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x844 | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_B: Simulation value -- Used for interrupts
15:0	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_A: Simulation value -- Used for interrupts

22.16.3.17 USB1_UTMIP_MISC_STS0_0

UTMIP MISC Status Register

Offset: 0x848 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
21	X	UTMIP_FS_DRV_EN: Indicates when the controller is driving on the bus
20:0	X	UTMIP_SPARE_FUSES: Spare Fuses value to keep the connections preserved

22.16.3.18 USB1_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup value

Offset: 0x84c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UTMIP_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UTMIP_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

22.16.3.19 USB1_UTMIP_BIAS_CFG2_0

UTMIP Bias cell configuration register 0

UTMIP REGISTER: UTMIP_BIAS_CFG2

Offset: 0x850 | Read/Write: R/W | Reset: 0x000180e0 (0bxxxxxxxxxxxxxx0011000000011100000)

Bit	Reset	Description
18:15	0x3	UTMIP_BIAS_SPARE: Spare bits for bias circuit
14:11	0x0	UTMIP_CHG_DIV: Apple charger divider reference control
10:9	0x0	UTMIP_TEST_SELECT: ATE test mode test tracking logic function
8:6	0x3	UTMIP_TEMP_COEFF: Bandgap temp coefficient control.
5:3	0x4	UTMIP_VREF_CTRL: Bandgap voltage control
2:0	0x0	UTMIP_HSSQUELCH_LEVEL_NEW: HS squelch detector level.

22.16.3.20 USB1_UTMIP_XCVR_CFG2_0

UTMIP transceiver cell configuration register 0

UTMIP REGISTER: UTMIP_XCVR_CFG2

Offset: 0x854 | Read/Write: R/W | Reset: 0x02200100 (0bxxxxxx10001000000000000100000000)

Bit	Reset	Description
25:22	0x8	UTMIP_XCVR_FSFSLEW_NEW: FS slew falling rate control.
21:18	0x8	UTMIP_XCVR_FSRSLEW_NEW: FS slew rising rate control.
17:14	0x0	UTMIP_XCVR_SPARE_NEW: spare pins for io pad
13:12	0x0	UTMIP_XCVR_VREG_DYN_DLY: Internal voltage regulator, voltage dynamic switching delay control
11:10	0x0	UTMIP_XCVR_VREG_LEV: Internal voltage regulator, control divided voltage level
9	0x0	UTMIP_XCVR_VREG_FIX18: Internal voltage regulator control to generate some divided voltage or follow VCLAMP.
8:5	0x8	UTMIP_PTERM_RANGE_ADJ: Termination range resistor select window.
4:2	0x0	UTMIP_XCVR_HSSLEW_NEW: HS slew rate control.
1:0	0x0	UTMIP_HS_COUP_EN: HighSpeed Ibias coupling control.

22.16.3.21 USB1_UTMIP_XCVR_CFG3_0

UTMIP transceiver cell configuration register 0

Offset: 0x858 | Read/Write: R/W | Reset: 0x00000066 (0bxxxxxx00000000000000000001100110)

Bit	Reset	Description
25:20	0x0	UTMIP_PCTRL_SW_VAL_NEW: PCTRL when software override is enabled
19:14	0x0	UTMIP_TCTRL_SW_VAL_NEW: TCTRL when software override is enabled
13:9	0x0	UTMIP_XCVR_RPD_CTRL: Active pull-down control signals
8	0x0	UTMIP_TERM_SEL: Auto termination enable.
7:4	0x6	UTMIP_XCVR_LSFSLEW_NEW: LS slew falling rate control.
3:0	0x6	UTMIP_XCVR_LSRSLEW_NEW: LS slew falling rate control.

22.16.3.22 USB2_QH_USB2D_QH_EP_n_OUT_0

USB2D Queue Head for OUT Endpoint n

There are 16 USB2D Queue Head for OUT Endpoint registers, where n = 0 through 15.

Offset: 0x1000 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint n. This is used to store a local Queue Head data structure for either Device Mode or Host Mode. In Device Mode, it holds the Queue Head for OUT endpoint n.

22.16.3.23 USB2_QH_USB2D_QH_EP_n_IN_0

USB2D Queue Head for IN Endpoint n

There are 16 USB2D Queue Head for IN Endpoint registers, where n = 0 through 15.

Offset: 0x1040 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint n. This is used to store a local Queue Head data structure for either Device Mode or Host Mode. In Device Mode, it holds the Queue Head for IN endpoint n.

22.16.4 USB2 Controller Registers

22.16.4.1 USB2_CONTROLLER_1_USB2D_ID_0

Base Address: USB2

USB2D Identification Register

Offset: 0x0 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One's complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

22.16.4.2 USB2_CONTROLLER_1_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for Host Mode. Set to 1.

22.16.4.3 USB2_CONTROLLER_1_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for Device Mode.

22.16.4.4 USB2_CONTROLLER_1_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

22.16.4.5 USB2_CONTROLLER_1_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words.
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

22.16.4.6 USB2_CONTROLLER_1_USB2D_GPTIMER0LD_0

The host/device controller drivers can measure time-related activities using these timer registers. These registers are not part of the standard EHCI controller.

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER0LD: This field has the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration.

22.16.4.7 USB2_CONTROLLER_1_USB2D_GPTIMER0CTRL_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit reloads the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt, and stops until the counter is reset by software. In repeat mode, the timer counts down to zero, generates an interrupt, and automatically reloads the counter to begin again.
23:0	RO	X	GPTCNT: This field has the value of the running timer.

22.16.4.8 USB2_CONTROLLER_1_USB2D_GPTIMER1LD_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:0	0x0	GPTIMER1LD: This field has the value to be loaded into the GPTCNT countdown timer on a reset action. The value in this register represents the time in microseconds minus 1 for the timer duration.

22.16.4.9 USB2_CONTROLLER_1_USB2D_GPTIMER1CTRL_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00XXXXXX (0b00xxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	GTPRUN: This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value.
30	WO	0x0	GPTRST: Writing a one to this bit reloads the GPTCNT with the value in GPTLD.
24	RW	0x0	GPTMODE: This bit selects between a single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt, and stops until the counter is reset by software. In repeat mode, the timer counts down to zero, generates an interrupt, and automatically reloads the counter to begin again.
23:0	RO	X	GPTCNT: This field has the value of the running timer.

22.16.4.10 USB2_CONTROLLER_1_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x30.

22.16.4.11 USB2_CONTROLLER_1_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

22.16.4.12 USB2_CONTROLLER_1_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: Indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: Indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: Indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

22.16.4.13 USB2_CONTROLLER_1_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.
17	X	LEN: Link Power Management Capability. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the USBCMD HIRD field, POSTSCx SSTS and DA fields, and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".
2	X	ASP: Asynchronous Schedule Park Capability. 1 = (Default) the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field is always 1.
1	X	PFL: Programmable Frame List Flag. 0 = System software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. 1 = System software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".

22.16.4.14 USB2_CONTROLLER_1_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

22.16.4.15 USB2_CONTROLLER_1_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2 0 host controller operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL,BA and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

22.16.4.16 USB2_CONTROLLER_1_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

22.16.4.17 USB2_CONTROLLER_1_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if the UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
1	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if the UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
0	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if the NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

22.16.4.18 USB2_CONTROLLER_1_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx00000000010000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50µs and each additional increment adds 75µs. For example, the value 0001b equals 125µs, and a value 1111b equals 1,175µs (~1.2ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.

Bit	R/W	Reset	Description
14	RW	0x0	ATDTW: Add DTD Tripwire. This bit is used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software and will be cleared by hardware when a hazard exists such that adding a dTD to a primed endpoint may go unnoticed. 0 = CLEAR 1 = SET
13	RW	0x0	SUTW: Setup Tripwire. This bit is used as a semaphore when the 8 bytes of setup data read extracted by the firmware. If the setup lockout mode is off, then there exists a hazard when new setup data arrives and firmware is copying setup data from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. 0 = CLEAR 1 = SET
11	RW	0x1	ASPE: Asynchronous Schedule Park mode Enable. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. 0 = DISABLE 1 = ENABLE
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation and is Read/Write capable.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL). Read Only. Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in Host Mode. Writing a one to this bit when Device Mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
4	RW	0x0	PSE: Periodic Schedule Enable. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size. (Read/Write). 000 = Default. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes) 100 = 64 elements (256 bytes) 101 = 32 elements (128 bytes) 110 = 16 elements (64 bytes) 111 = 8 elements (32 bytes) Only the host controller uses this field.

Bit	R/W	Reset	Description
1	RW	0x0	RST: Controller Reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller results in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit in Device Mode is not recommended. 0 = CLEAR 1 = SET
0	RW	0x0	RS: Run/Stop: Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HCHalted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event. 0 = STOP 1 = RUN

22.16.4.19 USB2_CONTROLLER_1_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule. 0= Disable Asynchronous Schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE: 1 = Periodic Schedule is enabled. 0 = Periodic Schedule is disabled. Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED

Bit	R/W	Reset	Description
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int). The alt_int bit is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY. The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to the connect and chirp. In Host Mode, this bit will be set every 125 µs and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller. 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also ad its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR

Bit	R/W	Reset	Description
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

22.16.4.20 USB2_CONTROLLER_1_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b0000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if the ULPI_ALT_INT bit in the USBSTS register transitions. The interrupt is acknowledged by software writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if the Interrupt on Async Advance bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if the Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

22.16.4.21 USB2_CONTROLLER_1_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in Host Mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In Device Mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode, bits 2:0 indicate the current micro-frame.

22.16.4.22 USB2_CONTROLLER_1_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

22.16.4.23 USB2_CONTROLLER_1_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

22.16.4.24 USB2_CONTROLLER_1_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bx0000000xxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
30:24	RW	0x0	TTHA: Internal TT Hub Address representation. This field is used to match the Hub Address field in QH (queue head) and siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address, then the packet will be broadcast on the High-Speed ports destined for a downstream High Speed hub with the address in QH/siTD.
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set). This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	RO	X	TTAS: Embedded TT Async Buffers Status. (Read Only). This read-only bit will be 1 if one or more transactions are being held in the embedded TT Async. Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

22.16.4.25 USB2_CONTROLLER_1_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

22.16.4.26 USB2_CONTROLLER_1_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in Host Mode before the packet begins on to the bus. The minimum value is 2, and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.

Bit	Reset	Description
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode

22.16.4.27 USB2_CONTROLLER_1_USB2D_ULPI_VIEWPORT_0

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Notes:

- *Writes to the ULPI through the viewport can substantially harm standard USB operations. Currently no usage model has been defined where software should need to execute writes directly to the ULPI PHY. See exception regarding optional features below.*
- *Executing read operations through the ULPI viewport should have no harmful side effects to standard USB operations.*

There are two operations that can be performed with the ULPI Viewport-- wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or Carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32 bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

USB2D ULPI Viewport Register

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., Carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

22.16.4.28 USB2_CONTROLLER_1_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in Host Mode. If in Device Mode it will be read only and always equal to 0000000b.
24:23	RO	X	SSTS: Suspend Status. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in Host Mode. If in Device Mode it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED 3 = PERIPH_NORESP_ERR

Bit	R/W	Reset	Description
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in Device Mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in Device Mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in Device Mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in Device Mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in Device Mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value Specific Test. 0000b Not enabled. 0001b J_ STATE. 0010b K_STATE. 0011b SEQ_NAK. 0100b Packet. 0101b FORCE_ENABLE. 0110b to 1111b Reserved. Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always be 0.
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED

Bit	R/W	Reset	Description
11:10	RO	X	LS: These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits is: 00b = SE0 10b = J-state 01b = K-state 11b = Undefined The value of this field is undefined if Port Power (PP) is zero in Host Mode. In Host Mode, the use of line state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In Device Mode, the use of line state by the device controller driver is not necessary. 0 = SE0 1 = K_STATE 2 = J_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When set to zero the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit, the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read-only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero), the results are undefined. This field is zero if Port Power (PP) is zero in Host Mode. In Device Mode: Read Only. This bit is a read-only status bit. 0 = NOT_SUSPEND 1 = SUSPEND

Bit	R/W	Reset	Description
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/ driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in Host Mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5 ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default) In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in Host Mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/ device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in Host Mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE

Bit	R/W	Reset	Description
0	RO	X	CCS: Current Connect Status. In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in Host Mode. In Device Mode: 1=Attached. 0=Not Attached (default). A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

22.16.4.29 USB2_CONTROLLER_1_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available when in Device Mode. This register shares the same address with the HOSTPC1 register (which is used only in Host Mode).

USB2D Host Mode LPM Behav and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available when in Host Mode. This register shares the same address with the DEVLC register (which is used only in Device Mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled, and every time the port enters the disconnect state, it will also enter in low power state, disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled (in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register (Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the register USBMODE 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED

Bit	R/W	Reset	Description
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Writing a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In Device Mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In Host Mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set - RW. Default = 00b. This bit field is valid during Host Mode Only. For Device Mode this is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power - RW. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode it is part of ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token - RW. Default = 0000b. This bit field is valid during Host Mode only. For Device Mode [19:18] is reserved and 17 is ASUS, while 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.
16	RW	0x0	STL: STALL reply to LPM token - RW. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode it is a part of ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT). 0 = DISABLE 1 = ENABLE
15:12	RW	0x0	LPMFRM: Auto LPM SOF Threshold - RW. Default = 0000b. This bit field is valid during Host Mode Only. For Device Mode this is reserved. This holds the SOF counter threshold. When the number of SOFs with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token is sent and the port enters the suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: bmAttributes - RO. Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.
0	RW	0x0	NYT_ASUS: NYET reply to LPM token - RW. Default = 0b. This bit is NYT during Device Mode. When this bit is '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. This bit is used to control the auto low power feature. If set, the auto low power feature is enabled and every time the port enters the suspend state, it also enters the low power state, disabling the transceiver clock. The behavior is the same as if the PHCD bit was enabled (this bit is set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

22.16.4.30 USB2_CONTROLLER_1_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx100x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0). PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET

Bit	R/W	Reset	Description
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x1	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: OTG Termination. This bit must be set when the OTG device is in Device Mode, this controls the pulldown on DM. 0 = NO_OTG_TERM 1 = OTG_TERM
1	RW	0x0	VC: VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP. 0 = NO_VBUS_CHRG 1 = VBUS_CHRG
0	RW	0x0	VD: VBUS_Discharge. Read/write. Setting this bit causes Vbus to discharge through a resistor. 0 = NO_VBUS_DISCHRG 1 = VBUS_DISCHRG

22.16.4.31 USB2_CONTROLLER_1_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay. Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in Host mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. Software should not use this.
4	RO	X	SDIS: Stream disable: 1 Streaming is disabled - helpful to avoid overruns/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In Device Mode, this bit controls the behavior of the setup lockout mechanism. 0 - Setup lockout is ON (default) 1 Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default]. 01 = Reserved. 10 = Device Controller. 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

22.16.4.32 USB2_CONTROLLER_1_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

22.16.4.33 USB2_CONTROLLER_1_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable Register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

22.16.4.34 USB2_CONTROLLER_1_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in Device Mode. 0 = NOT_RCVD 1 = SETUP_RCVD

22.16.4.35 USB2_CONTROLLER_1_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in Device Mode. 0 = DONT_PRIME 1 = PRIME

22.16.4.36 USB2_CONTROLLER_1_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in Device Mode. 0 = DONT_FLUSH 1 = FLUSH

22.16.4.37 USB2_CONTROLLER_1_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in Device Mode. 0 = NOT_READY 1 = READY

22.16.4.38 USB2_CONTROLLER_1_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in Device Mode. 0 = NOT_COMPLETE 1 = COMPLETE

22.16.4.39 USB2_CONTROLLER_1_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

22.16.4.40 USB2_CONTROLLER_1_USB2D_ENDPTCTRL<n>_0

USB2D Endpoint Control n Register

USB2D Endpoint Control 1 through 15 registers have the same field definitions and reset value. In the register offset, the value n is the register number (1 through 15).

Offset: 0x220 + (n – 1) * 0x04 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ

Bit	R/W	Reset	Description
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK 1 = EP_STALL

22.16.5 USB2 Controller Interface Registers

These are used to generate the actual RTL registers for USB2 controller interface.

ULPIS2S resets to be used as follows:

- Assert ULPIS2S_SLV0_CLAMP_XMIT to ensure that the signals between SLV0 and the line simulator are clean.
- Assert ULPIS2S_SLV0_RESET
- Do a functional asynchronous reset of the ChipIdea controller
- Deassert ULPIS2S_SLV0_RESET
- Deassert ULPIS2S_SLV0_CLAMP_XMIT

NULPI_SLV1_RESET should be used in the same way if there is some explicit way to reset the external ULPI controller (which is not necessarily the case). There is usually no reason to assert ULPIS2S_LINE_RESET.

Base Address: USB2

22.16.5.1 USB2_IF_USB_SUSP_CTRL_0

This register controls the suspend and resume behavior of the USB controller/PHY.

USB Suspend Control Register

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00111110000x1001000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1ms of spec. Used for Host mode ONLY. 0 = DISABLE 1 = ENABLE
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY Enabling this will only cutoff clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running. 0 = DISABLE 1 = ENABLE
24	RW	0x1	ULPI_PADS_CLKEN_RESET: Async reset for the synchronizers that are used in the external and loopback ULPI 60 MHz clock. 0 = DISABLE 1 = ENABLE
23	RW	0x1	ULPI_PADS_RESET: Async reset for trimmers and line state logic that is implemented in the pad macros. 0 = DISABLE 1 = ENABLE
22	RW	0x1	ULPIS2S_LINE_RESET: Async reset of the line simulator logic that sits between the two virtual PHYs (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
21	RW	0x1	ULPIS2S_SLV1_RESET: Async reset of the SLV1 ULPI logic. This corresponds to resetting the virtual PHY that is connected to the internal ULPI controller (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
20	RW	0x1	ULPIS2S_SLV0_RESET: Async reset of the SLV0 ULPI logic. This corresponds to resetting the virtual PHY that is connected to the external ULPI controller. (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
19	RW	0x0	UHSIC_PHY_ENB: Enable UHSIC PHY mode. Set this to 1 if using UHSIC PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
14	RW	0x1	UHSIC_RESET: Reset going to UHSIC PHY (active high). This should be set to 1 whenever programming the UHSIC config registers. It should be cleared to 0 after the programming of UHSIC config registers is done. UHSIC config registers should be programmed only once before doing any transactions on UHSIC. The UHSIC PHY registers should be programmed while UHSIC is in reset. 0 = DISABLE 1 = ENABLE
13	RW	0x0	ULPI_PHY_ENB: Enable ULPI PHY mode. Set this to 1 if using null or link ULPI PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on UTMIP. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to UHSIC PHY. 0 = Active low (default) 1 = Active high. This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (Device Mode). When enabled (1), USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in Device Mode, it is not applicable when USB controller is in Host Mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (Device Mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in Device Mode, it is not applicable when USB controller is in Host Mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, the USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB Host Modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable. If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status. This bit is set whenever the USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

22.16.5.2 USB2_IF_USB_ULPIS2S_CTRL_0

This register is used to set up parameters for ULPI null PHY mode.

> ***Note:*** *Current ULPI configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers*

ULPI NULL PHY Control Register

Offset: 0x418 | Read/Write: R/W | Reset: 0bxxxxxxxx0000xx0000000000xxxx0000

Bit	Reset	Description
23:20	0x0	ULPIS2S_CLAMP_LINE_DRIVE: The line drive value that should be sent into the line simulator during transmit clamping. The suggested value is 0x1: tri-state.
17	0x0	ULPIS2S_SLV1_CLAMP_XMIT: When set to 1, the outputs of the SLV1 transmit state machine are clamped to 0. This bit should be set before the SLV0 async reset is asserted to prevent meta-stability issues in the line simulator.
16	0x0	ULPIS2S_SLV0_CLAMP_XMIT: When set to 1, the outputs of the SLV0 transmit state machine are clamped to 0. This bit should be set before the SLV0 async reset is asserted to prevent meta-stability issues in the line simulator.
15	0x0	ULPIS2S_DISABLE_STP_PU: When set to 1 and in ULPIS2S mode, the pullup on the STP pin will NOT be active, even if the remote LINK asks to do so. In this case, an external pullup resistor would be required to ensure valid levels when the remote link is not powered.
14	0x0	ULPIS2S_SUPPORT_HS_KEEP_ALIVE: When enabled, the PHY will support HS KeepAlive packets. In that case, this would be the only thing that is supported in Opmode3. All other Opmode3 generate packets are not supported under any circumstances. 0 = DISABLE 1 = ENABLE
13	0x0	ULPIS2S_DISCON_DONT_CHECK_SE0: When enabled, the disconnect detection logic will only check that the other side is 'driving' tri-state. It will not check whether or not the local side is driving SE0. 0 = DISABLE 1 = ENABLE
12	0x0	ULPIS2S_FORCE_ULPI_CLK_OUT: When enabled and ULPIS2S_ENA is ENABLED, the external ULPI_CLOCK pad will always carry the internal 60 MHz clock, even if the interface is in shutdown mode. 0 = DISABLE 1 = ENABLE
11:8	0x0	ULPIS2S_SPARE: Reserved bits.
3	0x0	ULPIS2S_PLLU_MASTER_BLASTER60: When enabled, the PLLU 60 MHz clock will be forced on. 0 = DISABLE 1 = ENABLE
2	0x0	ULPIS2S_SUPPORT_DISCONNECT: When disabled, the PHY will never detect a Disconnect. 0 = DISABLE 1 = ENABLE
1	0x0	ULPIS2S_SLV1_FORCE_DEVICE: When disabled, the slave port that is connected to the pins can be programmed to be host or a device depending on the value of the DpPulldown and DmPulldown bits in the OTG_CTRL ULPI register. When enabled, the values of those bits in the OTG_CTRL register is ignored and the port will always behave like a device. 0 = DISABLE 1 = ENABLE
0	0x0	ULPIS2S_ENA: When enabled, the ULPI link interface coming out of the USB2 controller enters a NULL PHY with two slaves. As a result the external pins will have a slave ULPI interface. When disabled, the ULPI link interface coming out of the USB2 controller go straight to the pins. 0 = DISABLE 1 = ENABLE

22.16.5.3 USB2_IF_USB_ULPIS2S_SLV1_ID_0

This register controls the product and vendor ID fields for ULPI null PHY presented to external ULPI master.

Note: *Current ULPI configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers*

Offset: 0x41c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	ULPIS2S_SLV1_VENDOR_ID: PHY vendor_id as seen by the external ULPI master
15:0	0x0	ULPIS2S_SLV1_PRODUCT_ID: PHY product_id as seen by the external ULPI master

22.16.5.4 USB2_IF_USB_INTER_PKT_DELAY_CTRL_0

Inter Packet Delay Control

This register controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to Host Mode as device never transmits two packets in a row.

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for UHSIC PHY. Software should not change this.

22.16.5.5 USB2_IF_USB_RSM_DLY_0

USB Controller Resume Signaling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in Host Mode.

22.16.5.6 USB2_IF_SPARE_0

Note: *Software neither reads nor writes this register.*

For ICUSB PADCTLs

Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix SPARE_HI[1] - hsic_conn_det_feature_enable. Enable HSIC connect detection
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en. Enable HS_RESUME EOP fix SPARE_LO[5] - usb_port_suspend_fix_en

22.16.5.7 USB2_IF_ULPI_DIR_OVERRIDE_0

ULPI Override

Offset: 0x49c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1

Bit	Reset	Description
0	0x1	ULPI_DIR_OVERRIDE: By default, this bit is set. This will override ulpi_dir; i.e., when this bit is set, ulpi_dir is always asserted. Software needs to explicitly clear this bit, once slv1 Lp0 context is restored.

22.16.5.8 USB2_IF_USB2_NEW_CONTROL_0

USB Coherency and Memory Alignment Controls

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request.
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 style (0) and Tegra X1 style (1) DMA request generation mechanism. 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE

22.16.6 UHSIC Configuration Registers

Note: *Current HSIC configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers*

Base Address: USB2

22.16.6.1 USB2_UHSIC_MISC_STS0_0

UHSIC SPARE Fuse Value

Offset: 0xc30 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
0	X	UHSIC_TX_HS_VLD: Indicates when the controller is driving on the bus

22.16.6.2 USB2_UHSIC_PMC_WAKEUP0_0

UHSIC PMC Wakeup Value

Offset: 0xc34 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UHSIC_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UHSIC_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

22.16.6.3 USB2_UHSIC_UHSIC_PAD_TERM_0

UHSIC terminations

Offset: 0xc38 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4:0	X	UHSIC_RTERM

22.16.6.4 USB2_QH_USB2D_QH_EP_n_OUT_0

USB2D Queue Head for OUT Endpoint n

There are 16 USB2D Queue Head for OUT Endpoint registers, where n = 0 through 15.

Offset: 0x1000 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint n. This is used to store a local Queue Head data structure for either Device Mode or Host Mode. In Device Mode, it holds the Queue Head for OUT endpoint n.

22.16.6.5 USB2_QH_USB2D_QH_EP_n_IN_0

There are 16 USB2D Queue Head for IN Endpoint registers, where n = 0 through 15.

USB2D Queue Head for IN Endpoint n

Offset: 0x1040 + (n * 0x80) | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint n. This is used to store a local Queue Head data structure for either Device Mode or Host Mode. In Device Mode, it holds the Queue Head for IN endpoint n.

22.16.7 XUSB PADCTL Registers

Base Address: XUSB_PADCTL_BASE

22.16.7.1 XUSB_PADCTL_BOOT_MEDIA_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:1	0x0	BOOT_PORT: 0 = USB2_OTG0 1 = USB2_OTG1 2 = USB2_OTG2 3 = USB2_OTG3 4 = USB2_OTG4 5 = USB2_OTG5 6 = USB2_OTG6 9 = HSIC0 10 = HSIC1
0	0x0	BOOT_MEDIA_ENABLE: 0 = NO 1 = YES

22.16.7.2 XUSB_PADCTL_USB2_PAD_MUX_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000054 (0bxxxxxxxxxx00000000xxxxxx01010100)

Bit	Reset	Description
21	0x0	HSIC_PORT1_CONFIG: 0 = HSIC 1 = HSIC_PLUS

Bit	Reset	Description
20	0x0	HSIC_PORT0_CONFIG: 0 = HSIC 1 = HSIC_PLUS
19:18	0x0	USB2_BIAS_PAD: 0 = SNPS (USB2.0) 1 = XUSB 2 = UART
17:16	0x0	HSIC_PAD_TRK: 0 = SNPS (USB2.0) 1 = XUSB 2 = UART
15	0x0	USB2_HSIC_PAD_PORT1: 0 = SNPS (USB2.0) 1 = XUSB
14	0x0	USB2_HSIC_PAD_PORT0: 0 = SNPS (USB2.0) 1 = XUSB
7:6	0x1	USB2_OTG_PAD_PORT3: 0 = SNPS (USB2.0) 1 = XUSB 2 = UART
5:4	0x1	USB2_OTG_PAD_PORT2: 0 = SNPS (USB2.0) 1 = XUSB 2 = UART
3:2	0x1	USB2_OTG_PAD_PORT1: 0 = SNPS (USB2.0) 1 = XUSB 2 = UART
1:0	0x0	USB2_OTG_PAD_PORT0: 0 = SNPS (USB2.0) 1 = XUSB 2 = UART

22.16.7.3 XUSB_PADCTL_USB2_PORT_CAP_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	PORT3_REVERSE_ID: 0 = NO 1 = YES
14	0x0	PORT3_INTERNAL: 0 = NO 1 = YES
13:12	0x0	PORT3_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP
11	0x0	PORT2_REVERSE_ID: 0 = NO 1 = YES
10	0x0	PORT2_INTERNAL: 0 = NO 1 = YES
9:8	0x0	PORT2_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP

Bit	Reset	Description
7	0x0	PORT1_REVERSE_ID: 0 = NO 1 = YES
6	0x0	PORT1_INTERNAL: 0 = NO 1 = YES
5:4	0x0	PORT1_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP
3	0x0	PORT0_REVERSE_ID: 0 = NO 1 = YES
2	0x0	PORT0_INTERNAL: 0 = NO 1 = YES
1:0	0x0	PORT0_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP

22.16.7.4 XUSB_PADCTL_SNPS_OC_MAP_0

Offset: 0xc | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	CONTROLLER1_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_ DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED

22.16.7.5 XUSB_PADCTL_USB2_OC_MAP_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x0000ffff (0bxxxxxxxxxxxxxxxx1111111111111111)

Bit	Reset	Description
15:12	0xf	PORT3_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED
11:8	0xf	PORT2_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED

Bit	Reset	Description
7:4	0xf	PORT1_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED
3:0	0xf	PORT0_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED

22.16.7.6 XUSB_PADCTL_SS_PORT_MAP_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00039ce7 (0bxxxxxxxxxxxxx00111001110011100111)

Bit	Reset	Description
19	0x0	PORT3_INTERNAL: 0 = NO 1 = YES
18:15	0x7	PORT3_MAP: 0 = USB2_PORT0 1 = USB2_PORT1 2 = USB2_PORT2 3 = USB2_PORT3 7 = INIT_DISABLED
14	0x0	PORT2_INTERNAL: 0 = NO 1 = YES
13:10	0x7	PORT2_MAP: 0 = USB2_PORT0 1 = USB2_PORT1 2 = USB2_PORT2 3 = USB2_PORT3 7 = INIT_DISABLED
9	0x0	PORT1_INTERNAL: 0 = NO 1 = YES
8:5	0x7	PORT1_MAP: 0 = USB2_PORT0 1 = USB2_PORT1 2 = USB2_PORT2 3 = USB2_PORT3 7 = INIT_DISABLED
4	0x0	PORT0_INTERNAL: 0 = NO 1 = YES
3:0	0x7	PORT0_MAP: 0 = USB2_PORT0 1 = USB2_PORT1 2 = USB2_PORT2 3 = USB2_PORT3 7 = INIT_DISABLED

22.16.7.7 XUSB_PADCTL_VBUS_OC_MAP_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x000f7bde (0bxxxxxxxxxxxxx1111011101111011110)

Bit	Reset	Description
19:16	0xf	VBUS_ENABLE3_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED
15	0x0	VBUS_ENABLE3: 0 = NO 1 = YES
14:11	0xf	VBUS_ENABLE2_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED
10	0x0	VBUS_ENABLE2: 0 = NO 1 = YES
9:6	0xf	VBUS_ENABLE1_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED
5	0x0	VBUS_ENABLE1: 0 = NO 1 = YES
4:1	0xf	VBUS_ENABLE0_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 6 = OC_DETECTED_VBUS_PAD2 7 = OC_DETECTED_VBUS_PAD3 15 = OC_DETECTION_DISABLED
0	0x0	VBUS_ENABLE0: 0 = NO 1 = YES

22.16.7.8 XUSB_PADCTL_OC_DET_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000xxxx00000000xxxx0000)

Bit	Reset	Description
27	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD3: 0 = NO 1 = YES

Bit	Reset	Description
26	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD2: 0 = NO 1 = YES
25	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD1: 0 = NO 1 = YES
24	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD0: 0 = NO 1 = YES
23	0x0	OC_DETECTED_INTERRUPT_ENABLE3: 0 = NO 1 = YES
22	0x0	OC_DETECTED_INTERRUPT_ENABLE2: 0 = NO 1 = YES
21	0x0	OC_DETECTED_INTERRUPT_ENABLE1: 0 = NO 1 = YES
20	0x0	OC_DETECTED_INTERRUPT_ENABLE0: 0 = NO 1 = YES
15	0x0	OC_DETECTED_VBUS_PAD3: 0 = NO 1 = YES
14	0x0	OC_DETECTED_VBUS_PAD2: 0 = NO 1 = YES
13	0x0	OC_DETECTED_VBUS_PAD1: 0 = NO 1 = YES
12	0x0	OC_DETECTED_VBUS_PAD0: 0 = NO 1 = YES
11	0x0	OC_DETECTED3: 0 = NO 1 = YES
10	0x0	OC_DETECTED2: 0 = NO 1 = YES
9	0x0	OC_DETECTED1: 0 = NO 1 = YES
8	0x0	OC_DETECTED0: 0 = NO 1 = YES
3	0x0	SET_OC_DETECTED3: 0 = NO 1 = YES
2	0x0	SET_OC_DETECTED2: 0 = NO 1 = YES
1	0x0	SET_OC_DETECTED1: 0 = NO 1 = YES
0	0x0	SET_OC_DETECTED0: 0 = NO 1 = YES

22.16.7.9 XUSB_PADCTL_ELPG_PROGRAM_0_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b0000xxx0000xxx0000xxx0000xxx0000)

Bit	Reset	Description
31	0x0	USB2_HSIC_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
30	0x0	USB2_HSIC_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
29	0x0	USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
28	0x0	USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
24	0x0	SS_PORT3_WAKEUP_EVENT: 0 = NO 1 = YES
23	0x0	SS_PORT2_WAKEUP_EVENT: 0 = NO 1 = YES
22	0x0	SS_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
21	0x0	SS_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
17	0x0	SS_PORT3_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
16	0x0	SS_PORT2_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
15	0x0	SS_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
14	0x0	SS_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
10	0x0	USB2_PORT3_WAKEUP_EVENT: 0 = NO 1 = YES
9	0x0	USB2_PORT2_WAKEUP_EVENT: 0 = NO 1 = YES
8	0x0	USB2_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
7	0x0	USB2_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
3	0x0	USB2_PORT3_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
2	0x0	USB2_PORT2_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
1	0x0	USB2_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES

Bit	Reset	Description
0	0x0	USB2_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES

22.16.7.10 XUSB_PADCTL_ELPG_PROGRAM_1_0

Offset: 0x24 | Read/Write: R/W | Reset: 0xe0000fff (0b111xxxxxxxxxxxxxxxx111111111111)

Bit	Reset	Description
31	0x1	AUX_MUX_LP0_VCORE_DOWN: 0 = NO 1 = YES
30	0x1	AUX_MUX_LP0_CLAMP_EN_EARLY: 0 = NO 1 = YES
29	0x1	AUX_MUX_LP0_CLAMP_EN: 0 = NO 1 = YES
11	0x1	SSP3_ELPG_VCORE_DOWN: 0 = NO 1 = YES
10	0x1	SSP3_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
9	0x1	SSP3_ELPG_CLAMP_EN: 0 = NO 1 = YES
8	0x1	SSP2_ELPG_VCORE_DOWN: 0 = NO 1 = YES
7	0x1	SSP2_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
6	0x1	SSP2_ELPG_CLAMP_EN: 0 = NO 1 = YES
5	0x1	SSP1_ELPG_VCORE_DOWN: 0 = NO 1 = YES
4	0x1	SSP1_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
3	0x1	SSP1_ELPG_CLAMP_EN: 0 = NO 1 = YES
2	0x1	SSP0_ELPG_VCORE_DOWN: 0 = NO 1 = YES
1	0x1	SSP0_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
0	0x1	SSP0_ELPG_CLAMP_EN: 0 = NO 1 = YES

22.16.7.11 XUSB_PADCTL_USB3_PAD_MUX_0[1]

Offset: 0x28 | Read/Write: R/W | Reset: 0x817fc000 (0b10xxxx01011111111100xxx000000000)

Bit	Reset	Description
31:30	0x2	SATA_PAD_LANE0: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
25:24	0x1	PCIE_PAD_LANE6: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
23:22	0x1	PCIE_PAD_LANE5: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
21:20	0x3	PCIE_PAD_LANE4: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
19:18	0x3	PCIE_PAD_LANE3: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
17:16	0x3	PCIE_PAD_LANE2: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
15:14	0x3	PCIE_PAD_LANE1: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
13:12	0x0	PCIE_PAD_LANE0: 0 = PCIE_X1 1 = USB3_SS 2 = SATA 3 = PCIE_X4
8	0x0	FORCE_SATA_PAD_IDDQ_DISABLE_MASK0: 0 = NOT_DISABLED 1 = DISABLED
7	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK6: 0 = NOT_DISABLED 1 = DISABLED
6	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK5: 0 = NOT_DISABLED 1 = DISABLED
5	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4: 0 = NOT_DISABLED 1 = DISABLED
4	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3: 0 = NOT_DISABLED 1 = DISABLED
3	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2: 0 = NOT_DISABLED 1 = DISABLED

1. Not all lane configurations are supported, please refer to Section 22.7: USB PADCTL for a description of the supported configurations.

Bit	Reset	Description
2	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1: 0 = NOT_DISABLED 1 = DISABLED
1	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0: 0 = NOT_DISABLED 1 = DISABLED
0	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE: 0 = NOT_DISABLED 1 = DISABLED

22.16.7.12 XUSB_PADCTL_WAKE_CTRL_0

Wake logic AUX RDET CLK FORCE ON

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	LANE_S0_FORCE_TX_RDET_CLK_ENABLE
6	0x0	LANE_P6_FORCE_TX_RDET_CLK_ENABLE
5	0x0	LANE_P5_FORCE_TX_RDET_CLK_ENABLE
4	0x0	LANE_P4_FORCE_TX_RDET_CLK_ENABLE
3	0x0	LANE_P3_FORCE_TX_RDET_CLK_ENABLE
2	0x0	LANE_P2_FORCE_TX_RDET_CLK_ENABLE
1	0x0	LANE_P1_FORCE_TX_RDET_CLK_ENABLE
0	0x0	LANE_P0_FORCE_TX_RDET_CLK_ENABLE

22.16.7.13 XUSB_PADCTL_PM_SPARE_0

PAD MACRO SPARE BITS

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
11	0x0	HSIC_PM_SPARE_BIT3
10	0x0	HSIC_PM_SPARE_BIT2
9	0x0	HSIC_PM_SPARE_BIT1
8	0x0	HSIC_PM_SPARE_BIT0
3	0x0	OTG_PM_SPARE_BIT3
2	0x0	OTG_PM_SPARE_BIT2
1	0x0	OTG_PM_SPARE_BIT1
0	0x0	OTG_PM_SPARE_BIT0

22.16.7.14 XUSB_PADCTL_UPHY_CFG_STB_0

Setting to control the pulse width of the write/read data strobes for the UPHY pad cfg interface

Offset: 0x34 | Read/Write: R/W | Reset: 0x0000004d (0bxxxxxxxxxxxxxxxxxxxxx000001001101)

Bit	Reset	Description
11:6	0x1	ASSERT_DLY
5:0	0xd	PULSE_WIDTH

22.16.7.15 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL0_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000x000000000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

22.16.7.16 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_CTL1_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxx00000000xxxxx00000x0xxxx)

Bit	R/W	Reset	Description
23	RW	0x0	USBON_RPU_OVRD_VAL: 0 = NO 1 = YES
22	RW	0x0	USBON_RPU_OVRD: 0 = NO 1 = YES
21	RW	0x0	USBON_RPD_OVRD_VAL: 0 = NO 1 = YES
20	RW	0x0	USBON_RPD_OVRD: 0 = NO 1 = YES
19	RW	0x0	USBOP_RPU_OVRD_VAL: 0 = NO 1 = YES
18	RW	0x0	USBOP_RPU_OVRD: 0 = NO 1 = YES
17	RW	0x0	USBOP_RPD_OVRD_VAL: 0 = NO 1 = YES
16	RW	0x0	USBOP_RPD_OVRD: 0 = NO 1 = YES
10:9	RW	0x0	VREG_DYN_DLY
8:7	RW	0x0	VREG_LEV
6	RW	0x0	VREG_FIX18

Bit	R/W	Reset	Description
4	RW	0x0	DIV_DET_EN: 0 = NO 1 = YES
3	RO	X	VOP_DIV2P7_DET: 0 = NO 1 = YES
2	RO	X	VOP_DIV2P0_DET: 0 = NO 1 = YES
1	RO	X	VON_DIV2P7_DET: 0 = NO 1 = YES
0	RO	X	VON_DIV2P0_DET: 0 = NO 1 = YES

22.16.7.17 XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0

OTGPAD0,CTL0 static settings

Offset: 0x88 | Read/Write: R/W | Reset: 0x24cd1020 (0bxx10010011001101000100000010000)

Bit	Reset	Description
29	0x1	PD_ZI
28	0x0	PD2_OVRD_EN
27	0x0	PD2
26	0x1	PD
25	0x0	TERM_SEL
24:21	0x6	LS_FSLEW
20:17	0x6	LS_RSLEW
16:13	0x8	FS_FSLEW
12:9	0x8	FS_RSLEW
8:6	0x0	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

22.16.7.18 XUSB_PADCTL_USB2_OTG_PAD0_CTL_1_0

OTGPAD0,CTL1 static settings

Offset: 0x8c | Read/Write: R/W | Reset: 0x0X100044 (0bx00000x00001000000000000001000100)

Bit	R/W	Reset	Description
30:26	RW	0x0	RPD_CTRL
25	RO	X	RPU_STATUS_HIGH
24	RW	0x0	RPU_SWITCH_LOW
23	RW	0x0	RPU_SWITCH_OVRD
22	RW	0x0	HS_LOOPBACK_OVRD_VAL
21	RW	0x0	HS_LOOPBACK_OVRD_EN
20:17	RW	0x8	PTERM_RANGE_ADJ
16	RW	0x0	PD_DISC_OVRD_VAL
15	RW	0x0	PD_CHRP_OVRD_VAL
14:13	RW	0x0	RPU_RANGE_ADJ
12:11	RW	0x0	HS_COUP_EN
10:7	RW	0x0	SPARE
6:3	RW	0x8	TERM_RANGE_ADJ

Bit	R/W	Reset	Description
2	RW	0x1	PD_DR
1	RW	0x0	PD_DISC_OVRD
0	RW	0x0	PD_CHRP_OVRD

22.16.7.19 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD1_CTL0_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000xxxxx00000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

22.16.7.20 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD1_CTL1_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxx00000000xxxxx00000x0xxxx)

Bit	R/W	Reset	Description
23	RW	0x0	USBON_RPU_OVRD_VAL: 0 = NO 1 = YES
22	RW	0x0	USBON_RPU_OVRD: 0 = NO 1 = YES
21	RW	0x0	USBON_RPD_OVRD_VAL: 0 = NO 1 = YES
20	RW	0x0	USBON_RPD_OVRD: 0 = NO 1 = YES
19	RW	0x0	USBOP_RPU_OVRD_VAL: 0 = NO 1 = YES
18	RW	0x0	USBOP_RPU_OVRD: 0 = NO 1 = YES
17	RW	0x0	USBOP_RPD_OVRD_VAL: 0 = NO 1 = YES

Bit	R/W	Reset	Description
16	RW	0x0	USBOP_RPD_OVRD: 0 = NO 1 = YES
10:9	RW	0x0	VREG_DYN_DLY
8:7	RW	0x0	VREG_LEV
6	RW	0x0	VREG_FIX18
4	RW	0x0	DIV_DET_EN: 0 = NO 1 = YES
3	RO	X	VOP_DIV2P7_DET: 0 = NO 1 = YES
2	RO	X	VOP_DIV2P0_DET: 0 = NO 1 = YES
1	RO	X	VON_DIV2P7_DET: 0 = NO 1 = YES
0	RO	X	VON_DIV2P0_DET: 0 = NO 1 = YES

22.16.7.21 XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0

OTGPAD1,CTL0 static settings

Offset: 0xc8 | Read/Write: R/W | Reset: 0x24cd1020 (0bxx10010011001101000100000010000)

Bit	Reset	Description
29	0x1	PD_ZI
28	0x0	PD2_OVRD_EN
27	0x0	PD2
26	0x1	PD
25	0x0	TERM_SEL
24:21	0x6	LS_FSLEW
20:17	0x6	LS_RSLEW
16:13	0x8	FS_FSLEW
12:9	0x8	FS_RSLEW
8:6	0x0	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

22.16.7.22 XUSB_PADCTL_USB2_OTG_PAD1_CTL_1_0

OTGPAD1,CTL1 static settings

Offset: 0xcc | Read/Write: R/W | Reset: 0x0X100044 (0bx00000x00001000000000000001000100)

Bit	R/W	Reset	Description
30:26	RW	0x0	RPD_CTRL
25	RO	X	RPU_STATUS_HIGH
24	RW	0x0	RPU_SWITCH_LOW
23	RW	0x0	RPU_SWITCH_OVRD
22	RW	0x0	HS_LOOPBACK_OVRD_VAL
21	RW	0x0	HS_LOOPBACK_OVRD_EN
20:17	RW	0x8	PTERM_RANGE_ADJ

Bit	R/W	Reset	Description
16	RW	0x0	PD_DISC_OVRD_VAL
15	RW	0x0	PD_CHRP_OVRD_VAL
14:13	RW	0x0	RPU_RANGE_ADJ
12:11	RW	0x0	HS_COUP_EN
10:7	RW	0x0	SPARE
6:3	RW	0x8	TERM_RANGE_ADJ
2	RW	0x1	PD_DR
1	RW	0x0	PD_DISC_OVRD
0	RW	0x0	PD_CHRP_OVRD

22.16.7.23 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD2_CTL0_0

Offset: 0x100 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000xxxxx00000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES

Bit	R/W	Reset	Description
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

22.16.7.24 PADCTL_USB2_BATTERY_CHRG_OTGPAD2_CTL1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxx00000000xxxxx00000x0xxxx)

Bit	R/W	Reset	Description
23	RW	0x0	USBON_RPU_OVRD_VAL: 0 = NO 1 = YES
22	RW	0x0	USBON_RPU_OVRD: 0 = NO 1 = YES
21	RW	0x0	USBON_RPD_OVRD_VAL: 0 = NO 1 = YES
20	RW	0x0	USBON_RPD_OVRD: 0 = NO 1 = YES

Bit	R/W	Reset	Description
19	RW	0x0	USBOP_RPU_OVRD_VAL: 0 = NO 1 = YES
18	RW	0x0	USBOP_RPU_OVRD: 0 = NO 1 = YES
17	RW	0x0	USBOP_RPD_OVRD_VAL: 0 = NO 1 = YES
16	RW	0x0	USBOP_RPD_OVRD: 0 = NO 1 = YES
10:9	RW	0x0	VREG_DYN_DLY
8:7	RW	0x0	VREG_LEV
6	RW	0x0	VREG_FIX18
4	RW	0x0	DIV_DET_EN: 0 = NO 1 = YES
3	RO	X	VOP_DIV2P7_DET: 0 = NO 1 = YES
2	RO	X	VOP_DIV2P0_DET: 0 = NO 1 = YES
1	RO	X	VON_DIV2P7_DET: 0 = NO 1 = YES
0	RO	X	VON_DIV2P0_DET: 0 = NO 1 = YES

22.16.7.25 XUSB_PADCTL_USB2_OTG_PAD2_CTL_0_0

OTGPAD2 CTL0 static settings

Offset: 0x108 | Read/Write: R/W | Reset: 0x24cd1020 (0bxx10010011001101000100000010000)

Bit	Reset	Description
29	0x1	PD_ZI
28	0x0	PD2_OVRD_EN
27	0x0	PD2
26	0x1	PD
25	0x0	TERM_SEL
24:21	0x6	LS_FSLEW
20:17	0x6	LS_RSLEW
16:13	0x8	FS_FSLEW
12:9	0x8	FS_RSLEW
8:6	0x0	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

22.16.7.26 XUSB_PADCTL_USB2_OTG_PAD2_CTL_1_0

OTGPAD2 CTL1 static settings

Offset: 0x10c | Read/Write: R/W | Reset: 0x0X100044 (0bx00000x00001000000000000001000100)

22.16.7.27 XUSB_PADCTL_USB2_OTGPAD3_CTL_1_0

Bit	R/W	Reset	Description
30:26	RW	0x0	RPD_CTRL
25	RO	X	RPU_STATUS_HIGH
24	RW	0x0	RPU_SWITCH_LOW
23	RW	0x0	RPU_SWITCH_OVRD
22	RW	0x0	HS_LOOPBACK_OVRD_VAL
21	RW	0x0	HS_LOOPBACK_OVRD_EN
20:17	RW	0x8	PTERM_RANGE_ADJ
16	RW	0x0	PD_DISC_OVRD_VAL
15	RW	0x0	PD_CHRP_OVRD_VAL
14:13	RW	0x0	RPU_RANGE_ADJ
12:11	RW	0x0	HS_COUP_EN
10:7	RW	0x0	SPARE
6:3	RW	0x8	TERM_RANGE_ADJ
2	RW	0x1	PD_DR
1	RW	0x0	PD_DISC_OVRD
0	RW	0x0	PD_CHRP_OVRD

Offset: 0x140 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000xxxxx00000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

22.16.7.28 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD3_CTL1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxx00000000xxxxx00000x0xxxx)

Bit	R/W	Reset	Description
23	RW	0x0	USBON_RPU_OVRD_VAL: 0 = NO 1 = YES

Bit	R/W	Reset	Description
22	RW	0x0	USBON_RPU_OVRD: 0 = NO 1 = YES
21	RW	0x0	USBON_RPD_OVRD_VAL: 0 = NO 1 = YES
20	RW	0x0	USBON_RPD_OVRD: 0 = NO 1 = YES
19	RW	0x0	USBOP_RPU_OVRD_VAL: 0 = NO 1 = YES
18	RW	0x0	USBOP_RPU_OVRD: 0 = NO 1 = YES
17	RW	0x0	USBOP_RPD_OVRD_VAL: 0 = NO 1 = YES
16	RW	0x0	USBOP_RPD_OVRD: 0 = NO 1 = YES
10:9	RW	0x0	VREG_DYN_DLY
8:7	RW	0x0	VREG_LEV
6	RW	0x0	VREG_FIX18
4	RW	0x0	DIV_DET_EN: 0 = NO 1 = YES
3	RO	X	VOP_DIV2P7_DET: 0 = NO 1 = YES
2	RO	X	VOP_DIV2P0_DET: 0 = NO 1 = YES
1	RO	X	VON_DIV2P7_DET: 0 = NO 1 = YES
0	RO	X	VON_DIV2P0_DET: 0 = NO 1 = YES

22.16.7.29 XUSB_PADCTL_USB2_OTG_PAD3_CTL_0_0

OTGPAD3 CTL0 static settings

Offset: 0x148 | Read/Write: R/W | Reset: 0x24cd1020 (0bxx10010011001101000100000000100000)

Bit	Reset	Description
29	0x1	PD_ZI
28	0x0	PD2_OVRD_EN
27	0x0	PD2
26	0x1	PD
25	0x0	TERM_SEL
24:21	0x6	LS_FSLEW
20:17	0x6	LS_RSLEW
16:13	0x8	FS_FSLEW
12:9	0x8	FS_RSLEW
8:6	0x0	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

22.16.7.30 XUSB_PADCTL_USB2_OTG_PAD3_CTL_1_0

OTGPAD3_CTL1 static settings

Offset: 0x14c | Read/Write: R/W | Reset: 0x0X100044 (0bx00000x00001000000000000001000100)

Bit	R/W	Reset	Description
30:26	RW	0x0	RPD_CTRL
25	RO	X	RPU_STATUS_HIGH
24	RW	0x0	RPU_SWITCH_LOW
23	RW	0x0	RPU_SWITCH_OVRD
22	RW	0x0	HS_LOOPBACK_OVRD_VAL
21	RW	0x0	HS_LOOPBACK_OVRD_EN
20:17	RW	0x8	PTERM_RANGE_ADJ
16	RW	0x0	PD_DISC_OVRD_VAL
15	RW	0x0	PD_CHRP_OVRD_VAL
14:13	RW	0x0	RPU_RANGE_ADJ
12:11	RW	0x0	HS_COUP_EN
10:7	RW	0x0	SPARE
6:3	RW	0x8	TERM_RANGE_ADJ
2	RW	0x1	PD_DR
1	RW	0x0	PD_DISC_OVRD
0	RW	0x0	PD_CHRP_OVRD

22.16.7.31 XUSB_PADCTL_USB2_BATTERY_CHRG_TDCD_DBNC_TIMER_0

Offset: 0x280 | Read/Write: R/W | Reset: 0x00005000 (0bxxxxxxxxxxxxxxxx00101000000000000)

Bit	Reset	Description
16:11	0xa	IDDIG_DBNC
10:0	0x0	TDCD_DBNC

22.16.7.32 XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0

BIASPAD_CTL0 static settings

Offset: 0x284 | Read/Write: R/W | Reset: 0x260e0810 (0bxx10011000001110000010000000010000)

Bit	Reset	Description
29	0x1	TRK_PWR_ENA
28:25	0x3	SPARE
24:21	0x0	CHG_DIV
20:18	0x3	TEMP_COEF
17:15	0x4	VREF_CTRL
14:12	0x0	ADJRPU
11	0x1	PD
10:8	0x0	TERM_OFFSET
7:6	0x0	HS_CHIRP_LEVEL
5:3	0x2	HS_DISCON_LEVEL
2:0	0x0	HS_SQUELCH_LEVEL

22.16.7.33 XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0

These settings are used to trigger the tracking circuit in bias pad. De-assertion of PD_TRK triggers the FSM in hardware to enable the tracking in the pads. TRK_SW_OVRD can be used to completely disable the HW FSM and control the tracking circuit from software.

Offset: 0x288 | Read/Write: R/W | Reset: 0xX4820XXX (0bx00x0100100000100000xxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	FORCE_TRK_CLK_EN
29	RW	0x0	TRK_SW_OVRD
28	RO	X	TRK_DONE
27	RW	0x0	TRK_START
26	RW	0x1	PD_TRK
25:19	RW	0x10	TRK_DONE_RESET_TIMER
18:12	RW	0x20	TRK_START_TIMER
11:6	RO	X	PCTRL
5:0	RO	X	TCTRL

22.16.7.34 XUSB_PADCTL_HSIC_PAD0_CTL_0_0

HSIC PAD0 CTL0

Offset: 0x300 | Read/Write: R/W | Reset: 0x0000e3fe (0bxxxxxxxxxxxxxx0001110001111111110)

Bit	Reset	Description
18	0x0	RPU_STROBE
17	0x0	RPU_DATA1
16	0x0	RPU_DATA0
15	0x1	RPD_STROBE
14	0x1	RPD_DATA1
13	0x1	RPD_DATA0
12	0x0	LPBK_STROBE
11	0x0	LPBK_DATA1
10	0x0	LPBK_DATA0
9	0x1	PD_ZI_STROBE
8	0x1	PD_ZI_DATA1
7	0x1	PD_ZI_DATA0
6	0x1	PD_RX_STROBE
5	0x1	PD_RX_DATA1
4	0x1	PD_RX_DATA0
3	0x1	PD_TX_STROBE
2	0x1	PD_TX_DATA1
1	0x1	PD_TX_DATA0
0	0x0	IDDQ

22.16.7.35 XUSB_PADCTL_HSIC_PAD0_CTL_1_0

Offset: 0x304 | Read/Write: R/W | Reset: 0x00010008 (0bxxxxxxxxxxxxxxxx10000000000001000)

Bit	Reset	Description
16:12	0x10	RTERM
11:8	0x0	HSIC_OPT
7:4	0x0	TX_SLEW

Bit	Reset	Description
3:0	0x8	TX_RTUNEP

22.16.7.36 XUSB_PADCTL_HSIC_PAD0_CTL_2_0

Offset: 0x308 | Read/Write: R/W | Reset: 0x00000111 (0bxxxxxxxxxxxxxxxxxxxxx000100010001)

Bit	Reset	Description
11:8	0x1	RX_STROBE_TRIM
7:4	0x1	RX_DATA1_TRIM
3:0	0x1	RX_DATA0_TRIM

22.16.7.37 XUSB_PADCTL_HSIC_PAD1_CTL_0_0

HSIC PAD1 CTL0

Offset: 0x320 | Read/Write: R/W | Reset: 0x0000e3fe (0bxxxxxxxxxxxxxx0001110001111111110)

Bit	Reset	Description
18	0x0	RPU_STROBE
17	0x0	RPU_DATA1
16	0x0	RPU_DATA0
15	0x1	RPD_STROBE
14	0x1	RPD_DATA1
13	0x1	RPD_DATA0
12	0x0	LPBK_STROBE
11	0x0	LPBK_DATA1
10	0x0	LPBK_DATA0
9	0x1	PD_ZI_STROBE
8	0x1	PD_ZI_DATA1
7	0x1	PD_ZI_DATA0
6	0x1	PD_RX_STROBE
5	0x1	PD_RX_DATA1
4	0x1	PD_RX_DATA0
3	0x1	PD_TX_STROBE
2	0x1	PD_TX_DATA1
1	0x1	PD_TX_DATA0
0	0x0	IDDQ

22.16.7.38 XUSB_PADCTL_HSIC_PAD1_CTL_1_0

Offset: 0x324 | Read/Write: R/W | Reset: 0x00010008 (0bxxxxxxxxxxxxxxxx10000000000001000)

Bit	Reset	Description
16:12	0x10	RTERM
11:8	0x0	HSIC_OPT
7:4	0x0	TX_SLEW
3:0	0x8	TX_RTUNEP

22.16.7.39 XUSB_PADCTL_HSIC_PAD1_CTL_2_0

Offset: 0x328 | Read/Write: R/W | Reset: 0x00000111 (0bxxxxxxxxxxxxxxxxxxxxx000100010001)

Bit	Reset	Description
11:8	0x1	RX_STROBE_TRIM
7:4	0x1	RX_DATA1_TRIM
3:0	0x1	RX_DATA0_TRIM

22.16.7.40 XUSB_PADCTL_HSIC_PAD_TRK_CTL_0

These settings are used to trigger the tracking circuit in the HSIC pad. De-assertion of PD_TRK triggers the FSM in hardware to enable the tracking in the pads. TRK_SW_OVRD can be used to completely disable the hardware FSM and control the tracking circuit from software.

Offset: 0x340 | Read/Write: R/W | Reset: 0x01X904XX (0bxxxxxxx100x0100100000100000xxxxx)

Bit	R/W	Reset	Description
24	RW	0x1	AUTO_RTERM_EN
23	RW	0x0	FORCE_TRK_CLK_EN
22	RW	0x0	TRK_SW_OVRD
21	RO	X	TRK_DONE
20	RW	0x0	TRK_START
19	RW	0x1	PD_TRK
18:12	RW	0x10	TRK_DONE_RESET_TIMER
11:5	RW	0x20	TRK_START_TIMER
4:0	RO	X	RTERM_OUT

22.16.7.41 XUSB_PADCTL_HSIC_STRB_TRIM_CONTROL_0

HSIC STROBE trimmer control

Offset: 0x344 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000001)

Bit	Reset	Description
5:0	0x1	STRB_TRIM_VAL

22.16.7.42 XUSB_PADCTL_UPHY_PLL_P0_CTL_1_0

Start Registers for UPHY pads reserve space for 2 sets of PLL config registers. Each set has 32 registers, and 16 sets of lane control registers. Each set has a total of 16 registers.

PLL_CTL registers for PEX_USB PAD

Offset: 0x360 | Read/Write: R/W | Reset: 0x0190X017 (0bxx0000011001xx00xxxxx0000x10111)

Bit	R/W	Reset	Description
29:28	RW	0x0	PLL0_FREQ_PSDIV
27:20	RW	0x19	PLL0_FREQ_NDIV
17:16	RW	0x0	PLL0_FREQ_MDIV
15	RO	X	PLL0_LOCKDET_STATUS
9:8	RW	0x0	PLL0_MODE
7	RW	0x0	PLL0_BYPASS_EN
6	RW	0x0	PLL0_FREERUN_EN
4	RW	0x1	PLL0_PWR_OVRD
3	RW	0x0	PLL0_ENABLE

Bit	R/W	Reset	Description
2:1	RW	0x3	PLL0_SLEEP
0	RW	0x1	PLL0_IDDQ

22.16.7.43 XUSB_PADCTL_UPHY_PLL_P0_CTL_2_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x0000000X (0bxxxx00000000000000000000000000x0)

Bit	R/W	Reset	Description
27:4	RW	0x0	PLL0_CAL_CTRL
3	RW	0x0	PLL0_CAL_RESET
2	RW	0x0	PLL0_CAL_OVRD
1	RO	X	PLL0_CAL_DONE
0	RW	0x0	PLL0_CAL_EN

22.16.7.44 XUSB_PADCTL_UPHY_PLL_P0_CTL_3_0

Offset: 0x368 | Read/Write: R/W | Reset: 0x00037850 (0bxxxx00000000001101111000010 1xxx0)

Bit	Reset	Description
27:4	0x3785	PLL0_LOCKDET_CTRL
0	0x0	PLL0_LOCKDET_RESET

22.16.7.45 XUSB_PADCTL_UPHY_PLL_P0_CTL_4_0

Offset: 0x36c | Read/Write: R/W | Reset: 0x0050a100 (0bxxx0xxxx010100001x10xx010000xxx0)

Bit	Reset	Description
28	0x0	PLL0_TCLKOUT_EN
23:20	0x5	PLL0_CLKDIST_CTRL
19	0x0	PLL0_XDIGCLK_EN
18:16	0x0	PLL0_XDIGCLK_SEL
15	0x1	PLL0_TXCLKREF_EN
13:12	0x2	PLL0_TXCLKREF_SEL
9	0x0	PLL0_FBCLKBUF_EN
8	0x1	PLL0_REFCLKBUF_EN
7:4	0x0	PLL0_REFCLK_SEL
0	0x0	PLL0_REFCLK_TERM100

22.16.7.46 XUSB_PADCTL_UPHY_PLL_P0_CTL_5_0

Offset: 0x370 | Read/Write: R/W | Reset: 0x00000021 (0bxxxxxxxx00000000000000000010xx01)

Bit	Reset	Description
23:16	0x0	PLL0_DCO_CTRL
15:8	0x0	PLL0_LPF_CTRL
7:4	0x2	PLL0_CP_CTRL
1:0	0x1	PLL0_PFD_CTRL

22.16.7.47 XUSB_PADCTL_UPHY_PLL_P0_CTL_6_0

Offset: 0x374 | Read/Write: R/W | Reset: 0x02840000 (0b0x00x01010000100000000000000000)

Bit	Reset	Description
31	0x0	PLL0_FSEL_LOAD
29	0x0	PLL0_FSEL_COARSE_OVRD
28	0x0	PLL0_FSEL_FINE_OVRD
26:20	0x28	PLL0_FSEL_COARSE
19:0	0x40000	PLL0_FSEL_FINE

22.16.7.48 XUSB_PADCTL_UPHY_PLL_P0_CTL_7_0

Offset: 0x378 | Read/Write: R/W | Reset: 0x00000333 (0bxxxxxxxx00000000000000001100110011)

Bit	Reset	Description
23:0	0x333	PLL0_VREG_CTRL

22.16.7.49 XUSB_PADCTL_UPHY_PLL_P0_CTL_8_0

Offset: 0x37c | Read/Write: R/W | Reset: 0xXX070333 (0bxxxxxxxx0xx001110x00001100110011)

Bit	R/W	Reset	Description
31	RO	X	PLL0_RCAL_DONE
28:24	RO	X	PLL0_RCAL_VAL
23	RW	0x0	PLL0_RCAL_BYP_EN
20:16	RW	0x7	PLL0_RCAL_BYP_CODE
15	RW	0x0	PLL0_RCAL_OVRD
13	RW	0x0	PLL0_RCAL_CLK_EN
12	RW	0x0	PLL0_RCAL_EN
11:0	RW	0x333	PLL0_BGAP_CTRL

22.16.7.50 XUSB_PADCTL_UPHY_PLL_P0_CTL_9_0

Offset: 0x380 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PLL0_MISC_OUT
15:0	RW	0x0	PLL0_MISC_CTRL

22.16.7.51 XUSB_PADCTL_UPHY_PLL_P0_CTL_10_0

Offset: 0x384 | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	PLL0_CFG_RESET_
25	0x0	PLL0_CFG_RS
24	0x0	PLL0_CFG_WS
23:16	0x0	PLL0_CFG_ADDR
15:0	0x0	PLL0_CFG_WDATA

22.16.7.52 XUSB_PADCTL_UPHY_PLL_P0_CTL_11_0

Offset: 0x388 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	PLL0_CFG_RDATA

22.16.7.53 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_1_0

PAD_CTL registers for PEX_USB PAD

These include the Misc Registers to be used as overrides on functional controls or static/pseudo-static settings to be programmed through Miscellaneous registers irrespective of the pad lane owner

PEX_USB PAD LANE 0 Register

Offset: 0x460 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.54 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0

Offset: 0x464 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN

Bit	Reset	Description
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.55 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_3_0

Offset: 0x468 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.56 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_4_0

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.57 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_5_0

Offset: 0x470 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxx0xxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN

Bit	R/W	Reset	Description
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE
0	RW	0x0	RX_CAL_EN

22.16.7.58 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_6_0

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE
1:0	0x0	LOOP_NED_MODE

22.16.7.59 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_7_0

Offset: 0x478 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.60 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_8_0

Offset: 0x47c | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.61 XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_9_0

Offset: 0x480 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.62 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_1_0

PEX_USB PAD LANE 1 Register

Offset: 0x4a0 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP

Bit	R/W	Reset	Description
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.63 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.64 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_3_0

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN

Bit	R/W	Reset	Description
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.65 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_4_0

Offset: 0x4ac | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.66 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_5_0

Offset: 0x4b0 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxx0xxxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE
0	RW	0x0	RX_CAL_EN

22.16.7.67 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_6_0

Offset: 0x4b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE

Bit	Reset	Description
1:0	0x0	LOOP_NED_MODE

22.16.7.68 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_7_0

Offset: 0x4b8 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.69 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_8_0

Offset: 0x4bc | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.70 XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_9_0

Offset: 0x4c0 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.71 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_1_0

PEX_USB PAD LANE 2 Register

Offset: 0x4e0 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.72 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0

Offset: 0x4e4 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.73 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_3_0

Offset: 0x4e8 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.74 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_4_0

Offset: 0x4ec | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD

Bit	Reset	Description
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.75 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_5_0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxx0xxxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE
0	RW	0x0	RX_CAL_EN

22.16.7.76 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_6_0

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE
1:0	0x0	LOOP_NED_MODE

22.16.7.77 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_7_0

Offset: 0x4f8 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.78 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_8_0

Offset: 0x4fc | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.79 XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_9_0

Offset: 0x500 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.80 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_1_0

PEX_USB PAD LANE 3 Register

Offset: 0x520 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.81 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0

Offset: 0x524 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN

Bit	Reset	Description
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.82 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_3_0

Offset: 0x528 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.83 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_4_0

Offset: 0x52c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.84 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_5_0

Offset: 0x530 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxx0xxxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE

Bit	R/W	Reset	Description
0	RW	0x0	RX_CAL_EN

22.16.7.85 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_6_0

Offset: 0x534 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE
1:0	0x0	LOOP_NED_MODE

22.16.7.86 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_7_0

Offset: 0x538 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.87 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_8_0

Offset: 0x53c | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.88 XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_9_0

Offset: 0x540 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.89 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_1_0

PEX_USB PAD LANE 4 Register

Offset: 0x560 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE

Bit	R/W	Reset	Description
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.90 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0

Offset: 0x564 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.91 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_3_0

Offset: 0x568 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD

Bit	R/W	Reset	Description
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.92 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_4_0

Offset: 0x56c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.93 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_5_0

Offset: 0x570 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxx0xxxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE
0	RW	0x0	RX_CAL_EN

22.16.7.94 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_6_0

Offset: 0x574 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE
1:0	0x0	LOOP_NED_MODE

22.16.7.95 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_7_0

Offset: 0x578 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.96 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_8_0

Offset: 0x57c | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.97 XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_9_0

Offset: 0x580 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.98 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_1_0

PEX_USB PAD LANE 5 Register

Offset: 0x5a0 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.99 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_2_0

Offset: 0x5a4 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.100 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_3_0

Offset: 0x5a8 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.101 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_4_0

Offset: 0x5ac | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD

Bit	Reset	Description
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.102 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_5_0

Offset: 0x5b0 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxx0xxxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE
0	RW	0x0	RX_CAL_EN

22.16.7.103 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_6_0

Offset: 0x5b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE
1:0	0x0	LOOP_NED_MODE

22.16.7.104 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_7_0

Offset: 0x5b8 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.105 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_8_0

Offset: 0x5bc | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.106 XUSB_PADCTL_UPHY_MISC_PAD_P5_CTL_9_0

Offset: 0x5c0 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.107 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_1_0

PEX_USB PAD LANE 6 Register

Offset: 0x5e0 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.108 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_2_0

Offset: 0x5e4 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN

Bit	Reset	Description
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.109 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_3_0

Offset: 0x5e8 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.110 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_4_0

Offset: 0x5ec | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.111 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_5_0

Offset: 0x5f0 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxx0xxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE

Bit	R/W	Reset	Description
0	RW	0x0	RX_CAL_EN

22.16.7.112 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_6_0

Offset: 0x5f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE
1:0	0x0	LOOP_NED_MODE

22.16.7.113 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_7_0

Offset: 0x5f8 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.114 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_8_0

Offset: 0x5fc | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.115 XUSB_PADCTL_UPHY_MISC_PAD_P6_CTL_9_0

Offset: 0x600 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.116 XUSB_PADCTL_UPHY_PLL_S0_CTL_1_0

PLL_CTL registers for SATA_USB PAD

Offset: 0x860 | Read/Write: R/W | Reset: 0x01e0X017 (0bxx0000011110xx00xxxxx0000x10111)

Bit	R/W	Reset	Description
29:28	RW	0x0	PLL0_FREQ_PSDIV
27:20	RW	0x1e	PLL0_FREQ_NDIV
17:16	RW	0x0	PLL0_FREQ_MDIV
15	RO	X	PLL0_LOCKDET_STATUS
9:8	RW	0x0	PLL0_MODE
7	RW	0x0	PLL0_BYPASS_EN

Bit	R/W	Reset	Description
6	RW	0x0	PLL0_FREERUN_EN
4	RW	0x1	PLL0_PWR_OVRD
3	RW	0x0	PLL0_ENABLE
2:1	RW	0x3	PLL0_SLEEP
0	RW	0x1	PLL0_IDDQ

22.16.7.117 XUSB_PADCTL_UPHY_PLL_S0_CTL_2_0

Offset: 0x864 | Read/Write: R/W | Reset: 0x0000000X (0bxxxx00000000000000000000000000x0)

Bit	R/W	Reset	Description
27:4	RW	0x0	PLL0_CAL_CTRL
3	RW	0x0	PLL0_CAL_RESET
2	RW	0x0	PLL0_CAL_OVRD
1	RO	X	PLL0_CAL_DONE
0	RW	0x0	PLL0_CAL_EN

22.16.7.118 XUSB_PADCTL_UPHY_PLL_S0_CTL_3_0

Offset: 0x868 | Read/Write: R/W | Reset: 0x00037850 (0bxxxx00000000001101111000010 1xxx0)

Bit	Reset	Description
27:4	0x3785	PLL0_LOCKDET_CTRL
0	0x0	PLL0_LOCKDET_RESET

22.16.7.119 XUSB_PADCTL_UPHY_PLL_S0_CTL_4_0

Offset: 0x86c | Read/Write: R/W | Reset: 0x00508100 (0bxxx0xxxx010100001x00xx010000xxx0)

Bit	Reset	Description
28	0x0	PLL0_TCLKOUT_EN
23:20	0x5	PLL0_CLKDIST_CTRL
19	0x0	PLL0_XDIGCLK_EN
18:16	0x0	PLL0_XDIGCLK_SEL
15	0x1	PLL0_TXCLKREF_EN
13:12	0x0	PLL0_TXCLKREF_SEL
9	0x0	PLL0_FBCLKBUF_EN
8	0x1	PLL0_REFCLKBUF_EN
7:4	0x0	PLL0_REFCLK_SEL
0	0x0	PLL0_REFCLK_TERM100

22.16.7.120 XUSB_PADCTL_UPHY_PLL_S0_CTL_5_0

Offset: 0x870 | Read/Write: R/W | Reset: 0x00000021 (0bxxxxxxxx000000000000000010xx01)

Bit	Reset	Description
23:16	0x0	PLL0_DCO_CTRL
15:8	0x0	PLL0_LPF_CTRL
7:4	0x2	PLL0_CP_CTRL
1:0	0x1	PLL0_PFD_CTRL

22.16.7.121 XUSB_PADCTL_UPHY_PLL_S0_CTL_6_0

Offset: 0x874 | Read/Write: R/W | Reset: 0x04840000 (0b0x00x10010000100000000000000000)

Bit	Reset	Description
31	0x0	PLL0_FSEL_LOAD
29	0x0	PLL0_FSEL_COARSE_OVRD
28	0x0	PLL0_FSEL_FINE_OVRD
26:20	0x48	PLL0_FSEL_COARSE
19:0	0x40000	PLL0_FSEL_FINE

22.16.7.122 XUSB_PADCTL_UPHY_PLL_S0_CTL_7_0

Offset: 0x878 | Read/Write: R/W | Reset: 0x00000333 (0bxxxxxxxx00000000000000001100110011)

Bit	Reset	Description
23:0	0x333	PLL0_VREG_CTRL

22.16.7.123 XUSB_PADCTL_UPHY_PLL_S0_CTL_8_0

Offset: 0x87c | Read/Write: R/W | Reset: 0xXX070333 (0bxxxxxxxx0xx001110x00001100110011)

Bit	R/W	Reset	Description
31	RO	X	PLL0_RCAL_DONE
28:24	RO	X	PLL0_RCAL_VAL
23	RW	0x0	PLL0_RCAL_BYP_EN
20:16	RW	0x7	PLL0_RCAL_BYP_CODE
15	RW	0x0	PLL0_RCAL_OVRD
13	RW	0x0	PLL0_RCAL_CLK_EN
12	RW	0x0	PLL0_RCAL_EN
11:0	RW	0x333	PLL0_BGAP_CTRL

22.16.7.124 XUSB_PADCTL_UPHY_PLL_S0_CTL_9_0

Offset: 0x880 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PLL0_MISC_OUT
15:0	RW	0x0	PLL0_MISC_CTRL

22.16.7.125 XUSB_PADCTL_UPHY_PLL_S0_CTL_10_0

Offset: 0x884 | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	PLL0_CFG_RESET_
25	0x0	PLL0_CFG_RS
24	0x0	PLL0_CFG_WS
23:16	0x0	PLL0_CFG_ADDR
15:0	0x0	PLL0_CFG_WDATA

22.16.7.126 XUSB_PADCTL_UPHY_PLL_S0_CTL_11_0

Offset: 0x888 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	PLL0_CFG_RDATA

22.16.7.127 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_1_0

PAD_CTL registers for SATA_USB PAD

These include the Misc Registers to be used as overrides on functional controls or static/pseudo-static settings to be programmed through Misc registers irrespective of the pad lane owner.

SATA_USB PAD LANE 0 Register

Offset: 0x960 | Read/Write: R/W | Reset: 0xX00000X0 (0bxxxxx000x0000000xx00xxxxx000x000)

Bit	R/W	Reset	Description
29:28	RO	X	AUX_RX_STAT_IDLE
26	RW	0x0	AUX_RX_IDLE_BYP
25:24	RW	0x0	AUX_RX_IDLE_TH
22	RW	0x0	AUX_RX_IDLE_EN
21:20	RW	0x0	AUX_RX_IDLE_MODE
19	RW	0x0	AUX_RX_TERM_MODE
18	RW	0x0	AUX_RX_TERM_EN
17	RW	0x0	AUX_RX_IDDQ_OVRD
16	RW	0x0	AUX_RX_IDDQ
13	RW	0x0	AUX_RX_MODE_OVRD
12	RW	0x0	AUX_TX_MODE_OVRD
7	RO	X	AUX_TX_RDET_STATUS
6	RW	0x0	AUX_TX_RDET_CLK_EN
5	RW	0x0	AUX_TX_RDET_BYP
4	RW	0x0	AUX_TX_RDET_EN
2	RW	0x0	AUX_TX_TERM_EN
1	RW	0x0	AUX_TX_IDDQ_OVRD
0	RW	0x0	AUX_TX_IDDQ

22.16.7.128 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0

Offset: 0x964 | Read/Write: R/W | Reset: 0x00053030 (0b0000xx00000001010x11xx000011xx00)

Bit	Reset	Description
31	0x0	RX_RATE_PLL_OVRD
30	0x0	TX_RATE_PLL_OVRD
29	0x0	RX_RATE_PLL
28	0x0	TX_RATE_PLL
25	0x0	RX_PWR_OVRD
24	0x0	TX_PWR_OVRD
23:22	0x0	RX_RATE_PDIV
21:20	0x0	TX_RATE_PDIV
19:18	0x1	RX_RATE_SDIV
17:16	0x1	TX_RATE_SDIV
15	0x0	RX_DATA_EN

Bit	Reset	Description
13:12	0x3	RX_SLEEP
9	0x0	RX_IDDQ_OVRD
8	0x0	RX_IDDQ
7	0x0	TX_DATA_EN
6	0x0	TX_DATA_READY
5:4	0x3	TX_SLEEP
1	0x0	TX_IDDQ_OVRD
0	0x0	TX_IDDQ

22.16.7.129 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_3_0

Offset: 0x968 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxx000000000xxx0000xx00)

Bit	R/W	Reset	Description
19:16	RW	0x0	RX_BYP_CTRL
15	RW	0x0	RX_BYP_OVRD
14	RW	0x0	RX_BYP_EN
13:12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_REFCLK_EN
9:8	RO	X	RX_BYP_DATA
7	RW	0x0	TX_BYP_OVRD
6	RW	0x0	TX_BYP_EN
5:4	RW	0x0	TX_BYP_MODE
1:0	RW	0x0	TX_BYP_DATA

22.16.7.130 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_4_0

Offset: 0x96c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxx0x000xx0xxxx0000xxx00010)

Bit	Reset	Description
23	0x0	RX_TERM_OVRD
21	0x0	RX_TERM_EN
20	0x0	RX_TERM_MODE
19	0x0	TX_TERM_OVRD
16	0x0	TX_TERM_MODE
11	0x0	RX_CDR_RESET_OVRD
10	0x0	RX_CDR_RESET
9	0x0	TX_SYNC_OVRD
8	0x0	TX_SYNC
4	0x0	TX_SYNC_MODE
3:0	0x2	TX_SYNC_DLY

22.16.7.131 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_5_0

Offset: 0x970 | Read/Write: R/W | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxx0xxxxx00xx0xxx0)

Bit	R/W	Reset	Description
31:16	RO	X	RX_EOM_STATUS
15	RW	0x0	RX_CEE_OVRD
9	RO	X	RX_EOM_DONE
8	RW	0x0	RX_EOM_EN

Bit	R/W	Reset	Description
7	RW	0x0	RX_EQ_RESET
5	RO	X	RX_EQ_TRAIN_DONE
4	RW	0x0	RX_EQ_TRAIN_EN
1	RO	X	RX_CAL_DONE
0	RW	0x0	RX_CAL_EN

22.16.7.132 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_6_0

Offset: 0x974 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000xx000000xx00)

Bit	Reset	Description
15	0x0	LOOP_FED_EN
14	0x0	LOOP_FEA_EN
13	0x0	LOOP_NEA_EN
12	0x0	LOOP_NED_EN
9:8	0x0	LOOP_FED_MODE
7:4	0x0	LOOP_FEA_MODE
1:0	0x0	LOOP_NED_MODE

22.16.7.133 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_7_0

Offset: 0x978 | Read/Write: R/W | Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
23:16	RO	X	MISC_OUT
7:0	RW	0x0	MISC_CTRL

22.16.7.134 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_8_0

Offset: 0x97c | Read/Write: R/W | Reset: 0x08000000 (0bxxxx1x00000000000000000000000000)

Bit	Reset	Description
27	0x1	CFG_RESET_
25	0x0	CFG_RS
24	0x0	CFG_WS
23:16	0x0	CFG_ADDR
15:0	0x0	CFG_WDATA

22.16.7.135 XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_9_0

Offset: 0x980 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CFG_RDATA

22.16.7.136 XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_1_0

PAD_ECTL registers for USB3 ports

These include the Electrical control registers for the USB3 ports. These correspond to the pad interface signals which are MUXed per controller. This also includes the controls which are pseudo static for USB3.

Offset: 0xa60 | Read/Write: R/W | Reset: 0x0000001f (0bxxxx0000xxxx000000000000xx011111)

Bit	Reset	Description
27:24	0x0	RX_FELS
19:18	0x0	RX_TERM_CTRL
17:16	0x0	TX_TERM_CTRL
15:12	0x0	TX_DRV_CTRL
11:8	0x0	TX_DRV_SLEW
5:0	0x1f	TX_DRV_AMP

22.16.7.137 XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_2_0

Offset: 0xa64 | Read/Write: R/W | Reset: 0x0000008f (0bxxxxxxxxxxxxx00000000000010001111)

Bit	Reset	Description
19:16	0x0	RX_IQ_CTRL
15:0	0x8f	RX_CTLE

22.16.7.138 XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_3_0

Offset: 0xa68 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_DFE

22.16.7.139 XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_4_0

Offset: 0xa6c | Read/Write: R/W | Reset: 0x01c70000 (0b00000001110001111xxxxxxx00000000)

Bit	Reset	Description
31:16	0x1c7	RX_CDR_CTRL
7:0	0x0	RX_PI_CTRL

22.16.7.140 XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_5_0

Offset: 0xa70 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_L

22.16.7.141 XUSB_PADCTL_UPHY_USB3_PAD0_ECTL_6_0

Offset: 0xa74 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_H

22.16.7.142 XUSB_PADCTL_UPHY_USB3_PAD0_CTL_0_0

Offset: 0xa78 | Read/Write: R/W | Reset: 0x00000404 (0bxxxxxxxxxxxxxxxxxxxx00100xxx00100)

Bit	Reset	Description
12:11	0x0	RX_RATE_PDIV
10:9	0x2	RX_RATE_SDIV
8	0x0	RX_RATE_PLL
4:3	0x0	TX_RATE_PDIV

Bit	Reset	Description
2:1	0x2	TX_RATE_SDIV
0	0x0	TX_RATE_PLL

22.16.7.143 XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_1_0

PAD_ECTL Port 1 Register

Offset: 0xaa0 | Read/Write: R/W | Reset: 0x0000001f (0bxxxx0000xxxx000000000000xx011111)

Bit	Reset	Description
27:24	0x0	RX_FELS
19:18	0x0	RX_TERM_CTRL
17:16	0x0	TX_TERM_CTRL
15:12	0x0	TX_DRV_CTRL
11:8	0x0	TX_DRV_SLEW
5:0	0x1f	TX_DRV_AMP

22.16.7.144 XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_2_0

Offset: 0xaa4 | Read/Write: R/W | Reset: 0x0000008f (0bxxxxxxxxxxxxx0000000000010001111)

Bit	Reset	Description
19:16	0x0	RX_IQ_CTRL
15:0	0x8f	RX_CTLE

22.16.7.145 XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_3_0

Offset: 0xaa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_DFE

22.16.7.146 XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_4_0

Offset: 0xaac | Read/Write: R/W | Reset: 0x01c70000 (0b00000001110001111xxxxxxx00000000)

Bit	Reset	Description
31:16	0x1c7	RX_CDR_CTRL
7:0	0x0	RX_PI_CTRL

22.16.7.147 XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_5_0

Offset: 0xab0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_L

22.16.7.148 XUSB_PADCTL_UPHY_USB3_PAD1_ECTL_6_0

Offset: 0xab4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_H

22.16.7.149 XUSB_PADCTL_UPHY_USB3_PAD1_CTL_0_0

Offset: 0xab8 | Read/Write: R/W | Reset: 0x00000404 (0bxxxxxxxxxxxxxxxxxxxx00100xxx00100)

Bit	Reset	Description
12:11	0x0	RX_RATE_PDIV
10:9	0x2	RX_RATE_SDIV
8	0x0	RX_RATE_PLL
4:3	0x0	TX_RATE_PDIV
2:1	0x2	TX_RATE_SDIV
0	0x0	TX_RATE_PLL

22.16.7.150 XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_1_0

PAD_ECTL Port 2 Register

Offset: 0xae0 | Read/Write: R/W | Reset: 0x0000001f (0bxxxx0000xxxx000000000000xx011111)

Bit	Reset	Description
27:24	0x0	RX_FELS
19:18	0x0	RX_TERM_CTRL
17:16	0x0	TX_TERM_CTRL
15:12	0x0	TX_DRV_CTRL
11:8	0x0	TX_DRV_SLEW
5:0	0x1f	TX_DRV_AMP

22.16.7.151 XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_2_0

Offset: 0xae4 | Read/Write: R/W | Reset: 0x0000008f (0bxxxxxxxxxxxxx00000000000010001111)

Bit	Reset	Description
19:16	0x0	RX_IQ_CTRL
15:0	0x8f	RX_CTLE

22.16.7.152 XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_3_0

Offset: 0xae8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_DFE

22.16.7.153 XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_4_0

Offset: 0xaec | Read/Write: R/W | Reset: 0x01c70000 (0b00000001110001111xxxxxxx00000000)

Bit	Reset	Description
31:16	0x1c7	RX_CDR_CTRL
7:0	0x0	RX_PI_CTRL

22.16.7.154 XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_5_0

Offset: 0xaf0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_L

22.16.7.155 XUSB_PADCTL_UPHY_USB3_PAD2_ECTL_6_0

Offset: 0xaf4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_H

22.16.7.156 XUSB_PADCTL_UPHY_USB3_PAD2_CTL_0_0

Offset: 0xaf8 | Read/Write: R/W | Reset: 0x00000404 (0bxxxxxxxxxxxxxxxxxxxx00100xxx00100)

Bit	Reset	Description
12:11	0x0	RX_RATE_PDIV
10:9	0x2	RX_RATE_SDIV
8	0x0	RX_RATE_PLL
4:3	0x0	TX_RATE_PDIV
2:1	0x2	TX_RATE_SDIV
0	0x0	TX_RATE_PLL

22.16.7.157 XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_1_0

PAD_ECTL Port 3 Register

Offset: 0xb20 | Read/Write: R/W | Reset: 0x0000001f (0bxxxx0000xxxx000000000000xx011111)

Bit	Reset	Description
27:24	0x0	RX_FELS
19:18	0x0	RX_TERM_CTRL
17:16	0x0	TX_TERM_CTRL
15:12	0x0	TX_DRV_CTRL
11:8	0x0	TX_DRV_SLEW
5:0	0x1f	TX_DRV_AMP

22.16.7.158 XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_2_0

Offset: 0xb24 | Read/Write: R/W | Reset: 0x0000008f (0bxxxxxxxxxxxxx00000000000010001111)

Bit	Reset	Description
19:16	0x0	RX_IQ_CTRL
15:0	0x8f	RX_CTLE

22.16.7.159 XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_3_0

Offset: 0xb28 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_DFE

22.16.7.160 XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_4_0

Offset: 0xb2c | Read/Write: R/W | Reset: 0x01c70000 (0b00000001110001111xxxxxxx00000000)

Bit	Reset	Description
31:16	0x1c7	RX_CDR_CTRL
7:0	0x0	RX_PI_CTRL

22.16.7.161 XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_5_0

Offset: 0xb30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_L

22.16.7.162 XUSB_PADCTL_UPHY_USB3_PAD3_ECTL_6_0

Offset: 0xb34 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RX_EQ_CTRL_H

22.16.7.163 XUSB_PADCTL_UPHY_USB3_PAD3_CTL_0_0

Offset: 0xb38 | Read/Write: R/W | Reset: 0x00000404 (0bxxxxxxxxxxxxxxxxxxxx00100xxx00100)

Bit	Reset	Description
12:11	0x0	RX_RATE_PDIV
10:9	0x2	RX_RATE_SDIV
8	0x0	RX_RATE_PLL
4:3	0x0	TX_RATE_PDIV
2:1	0x2	TX_RATE_SDIV
0	0x0	TX_RATE_PLL

22.16.7.164 XUSB_PADCTL_USB2_VBUS_ID_0

VBUS and ID status and override for Device Mode/OTG port. Placing register towards the end of the PADCTL space

ID_OVERRIDE is 4 bits [1D,ID_A,ID_B,ID_C]

Offset: 0xc60 | Read/Write: R/W | Reset: 0x00X11XXX (0bxxxxxxxx00x100001000100xxxx00x00x)

Bit	R/W	Reset	Description
24	RW	0x0	VBUS_WAKEUP_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	VBUS_WAKEUP_ST_CHNG: 0 = NO 1 = YES
22	RO	X	VBUS_WAKEUP: 0 = NO 1 = YES
21:18	RW	0x8	ID_OVERRIDE: 0 = ID_GND 4 = ID_A 2 = ID_B 1 = ID_C 8 = ID_FLOAT
17:16	RW	0x1	ID_SOURCE_SELECT: 0 = VGPIO 1 = ID_OVERRIDE
15	RW	0x0	VBUS_WAKEUP_OVERRIDE: 0 = NO 1 = YES

Bit	R/W	Reset	Description
14	RW	0x0	VBUS_OVERRIDE: 0 = NO 1 = YES 1 = VBUS_ON 0 = VBUS_OFF
13:12	RW	0x1	VBUS_SOURCE_SELECT: 0 = VGPIO 1 = VBUS_OVERRIDE
11	RW	0x0	IDDIG_CHNG_INTR_EN: 0 = NO 1 = YES
10	RW	0x0	IDDIG_ST_CHNG: 0 = NO 1 = YES
9	RO	X	IDDIG_C: 0 = NO 1 = YES
8	RO	X	IDDIG_B: 0 = NO 1 = YES
7	RO	X	IDDIG_A: 0 = NO 1 = YES
6	RO	X	IDDIG: 0 = NO 1 = YES
5	RW	0x0	VBUS_VALID_CHNG_INTR_EN: 0 = NO 1 = YES
4	RW	0x0	VBUS_VALID_ST_CHNG: 0 = NO 1 = YES
3	RO	X	VBUS_VALID: 0 = NO 1 = YES
2	RW	0x0	OTG_VBUS_SESS_VLD_CHNG_INTR_EN: 0 = NO 1 = YES
1	RW	0x0	OTG_VBUS_SESS_VLD_ST_CHNG: 0 = NO 1 = YES
0	RO	X	OTG_VBUS_SESS_VLD: 0 = NO 1 = YES

22.16.8 XUSB Host IPFS Registers

Base Address: XUSB_HOST_BASE + 0x9000 (0x70099000)

22.16.8.1 XUSB_HOST_AXI_BAR0_SZ_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxx00000000000000001000)

Bit	Reset	Description
19:0	0x8	AXI_BAR0_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.8.2 XUSB_HOST_AXI_BAR1_SZ_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR1_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.8.3 XUSB_HOST_AXI_BAR2_SZ_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR2_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.8.4 XUSB_HOST_AXI_BAR3_SZ_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR3_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.8.5 XUSB_HOST_AXI_BAR0_START_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x70090000 (0b01110000000010010000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x70090	AXI_BAR0_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.8.6 XUSB_HOST_AXI_BAR1_START_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR1_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.8.7 XUSB_HOST_AXI_BAR2_START_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR2_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.8.8 XUSB_HOST_AXI_BAR3_START_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR3_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.8.9 XUSB_HOST_FPCI_BAR0_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00700901 (0b00000000011100000000010010000xxx1)

Bit	Reset	Description
31:4	0x70090	FPCI_BAR0_START: The start of the FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR0_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = memory-mapped access (PW only) 1 = I/O or config access (NPW only)

22.16.8.10 XUSB_HOST_FPCI_BAR1_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR1_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR1_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

22.16.8.11 XUSB_HOST_FPCI_BAR2_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR2_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR2_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

22.16.8.12 XUSB_HOST_FPCI_BAR3_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR3_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR3_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

22.16.8.13 XUSB_HOST_MSI_BAR_SZ_0

MSI BAR SIZE

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE: The size of the address range associated with MSI BAR is in 4K increments. Value of 0 signifies BAR is not used.

22.16.8.14 XUSB_HOST_MSI_AXI_BAR_ST_0

MSI AXI BAR START

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	MSI_AXI_BAR_START: The start of the upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for the MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

22.16.8.15 XUSB_HOST_MSI_FPCI_BAR_ST_0

MSI FPCI BAR START

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xxxx)

Bit	Reset	Description
31:4	0x0	MSI_FPCI_BAR_START: The start of the upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

22.16.8.16 XUSB_HOST_MSI_VEC0_0

MSI VECTORi i in [0 7] RW

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR0: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.17 XUSB_HOST_MSI_VEC1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR1: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.18 XUSB_HOST_MSI_VEC2_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR2: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.19 XUSB_HOST_MSI_VEC3_0

Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR3: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.20 XUSB_HOST_MSI_VEC4_0

Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR4: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.21 XUSB_HOST_MSI_VEC5_0

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR5: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.22 XUSB_HOST_MSI_VEC6_0

Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR6: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.23 XUSB_HOST_MSI_VEC7_0

Offset: 0x11c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR7: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.8.24 XUSB_HOST_MSI_EN_VEC0_0

MSI ENABLE VECTORi□i in [0□7]□RW

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR0: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.25 XUSB_HOST_MSI_EN_VEC1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR1: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.26 XUSB_HOST_MSI_EN_VEC2_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR2: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.27 XUSB_HOST_MSI_EN_VEC3_0

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR3: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.28 XUSB_HOST_MSI_EN_VEC4_0

Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR4: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.29 XUSB_HOST_MSI_EN_VEC5_0

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR5: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.30 XUSB_HOST_MSI_EN_VEC6_0

Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR6: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.31 XUSB_HOST_MSI_EN_VEC7_0

Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR7: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.8.32 XUSB_HOST_CONFIGURATION_0

Configuration

Offset: 0x180 | Read/Write: R/W | Reset: 0x800X8X40 (0b1xxxxxxxxxxx1xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x1	CLKEN_OVERRIDE: This can override the clock enable in case of a malfunction.
19	RW	0x1	PW_NO_DEVSEL_ERR_CYA: Setting this bit disables detection of DECERR due to no DEVSEL for DS PWs only.
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on AFI upstream A value of 1b indicates there are no outstanding reads to the initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on AFI upstream A value of 1b indicates there are no outstanding writes to the initiator.
15	RW	0x1	WDATA_LEAD_CYA: Used to enable/disable the handling of write data ahead of requests on IPFS AXI target.
14	RW	0x0	WR_INTRLV_CYA: Used to enable/disable the handling of interleaved write requests on IPFS AXI target.
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to IPFS target. A value of 1b indicates there are no outstanding reads to the downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to IPFS target. A value of 1b indicates there are no outstanding writes to the downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any active bits valid or not.
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = Whenever MSI is ready assert the interrupt 0 = Default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: Input to the upstream FPCI 1 = Whenever write is ready, send it. 0 = Write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to the upstream FPCI. Allows the upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for the upstream FPCI. Allows the upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: Used for the downstream FPCI. Allows the downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: Input to the downstream FPCI. Allows the downstream FPCI responses to pass writes
1	RW	0x0	DFPCI_PASSPW: Input to the downstream FPCI. Allow the downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the IPFS device block is disabled, it is completely invisible on the IPFS bus; i.e., it does not even process IPFS configuration accesses.

22.16.8.33 XUSB_HOST_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows an FPCI error to be forwarded to an AXI response when the FPCI error response indicates a Master Abort. 1 = Forward error 0 = Return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows an FPCI error to be forwarded to an AXI response when the FPCI error response indicates a Data Error. 1 = Forward error 0 = Return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows an FPCI error to be forwarded to an AXI response when FPCI error response indicates a Target Abort. This bit also covers a decode error generated when there is no DEVSEL received. 1 = Forward error 0 = Return AXI OKAY response (2'b0)

22.16.8.34 XUSB_HOST_INTR_MASK_0

Interrupt Masks

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxx0xxxxxxx0)

Bit	Reset	Description
16	0x0	IP_INT_MASK: IP (SATA/AZA) interrupt to the CPU complex gated by the mask.
8	0x0	MSI_MASK: MSI to the CPU complex gated by the mask.
0	0x0	INT_MASK: Interrupt to the CPU complex gated by the mask.

22.16.8.35 XUSB_HOST_INTR_CODE_0

Interrupt Control

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: Eight interrupt codes. If the code is 0, logging of the next interrupt is enabled 0 = INT_CODE_CLEAR: Clear interrupt code 1 = INT_CODE_INI_SLVERR: Interrupt code for CPU AXI SLVERR response to IPFS 2 = INT_CODE_INI_DECERR: Interrupt code for CPU AXI DECERR response to IPFS 3 = INT_CODE_TGT_SLVERR: Interrupt code for PCIe endpoint FPCI target abort or data error response to IPFS 4 = INT_CODE_TGT_DECERR: Interrupt code for PCIe2 FPCI master abort response to IPFS 5 = INT_CODE_TGT_WRERR: Interrupt code for bufferable write to non-posted write address region 6 = RSVD1: Reserved 7 = INT_CODE_DFPCI_DECERR: Interrupt code for PCIe2 response to downstream request when downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR: Interrupt code for IPFS response to downstream request when AXI target AXI address does not fall in any IPFS downstream BARs 9 = INT_CODE_FPCI_TIMEOUT: Interrupt code for FPCI Timeout 10 = RSVD2: Reserved for future expansion 11 = RSVD3 12 = RSVD4 13 = RSVD5 14 = RSVD6 15 = INT_CODE_SM_FATAL_ERROR: Interrupt code for SM fatal error 16 = INT_CODE_SM_NON_FATAL_ERROR: Interrupt code for SM non-fatal error

22.16.8.36 XUSB_HOST_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, this field contains address bits [31:2], either in FPCI memory space or AXI space. For FPCI generated errors, the field contains the FPCI address. For AXI/IPFS generated errors, the field contains the AXI address.
0	0x0	DIR: Indicates the direction of the AXI/FPCI transaction. 1=RD/0=WRIF signature type is 6 (sideband message), this field is 1. 0 = WRITE: Interrupt due to a write transaction 1 = READ: Interrupt due to a read transaction

22.16.8.37 XUSB_HOST_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of the captured FPCI address (bits [39:32]) when the interrupt code is 3, 4, or 7. These bits determine the region in the Hypertransport Address Map that was accessed.

22.16.8.38 XUSB_HOST_IPFS_INTR_ENABLE_0

IPFS Interrupt Enable

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxx00000000)

Bit	Reset	Description
13	0x0	EN_SM_NON_FATAL_ERROR: Enable bit for interrupt code 15
12	0x0	EN_SM_FATAL_ERROR: Enable bit for interrupt code 14
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

22.16.8.39 XUSB_HOST_UFPCI_CONFIG_0

Upstream FPCI Configuration

Offset: 0x19c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00010)

Bit	Reset	Description
4:0	0x2	UNITID_T0C0: Upstream FPCI Unit ID for controller 0. HyperTransport, upstream FPCI request

22.16.8.40 XUSB_HOST_CFG_REVID_0

CFG_REVID Register

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x000X10XX (0bxxxxxxxxxxxxxxxxxx0100xxxxxxx1xx)

Bit	R/W	Reset	Description
19	RO	X	DEV2SM_NONISO_REQUEST_PEND: This bit indicates if there is a non-ISO request pending. 0 = NO 1 = YES
18	RO	X	DEV2SM_ISO_REQUEST_PEND: This bit indicates if there is an ISO request pending. 0 = NO 1 = YES
13:12	RW	0x1	STRAP_CPU_MODE: MCP: Mode to send MSI. It can be programmable. 0 = NB_INTEL 1 = NB_AMD 2 = AMD 3 = TMTA
11	RW	0x0	CFG_REVID_WRITE_ENABLE: MCP: The enable to override the rev ID. It can be programmable. 0 = CLEAR 1 = SET
10	RW	0x0	CFG_REVID_OVERRIDE: MCP: Provides a way to override the current revision ID. It can be programmable. 0 = DISABLE 1 = ENABLE
4	RO	X	DEV2LEG_NONCOH_REQUEST_PEND: MCP: Tells the leg block that a non-coherent request is pending. 0 = NO 1 = YES
3	RO	X	DEV2LEG_COH_REQUEST_PEND: MCP comment: Tells the leg block that a coherent request is pending. 0 = NO 1 = YES

Bit	R/W	Reset	Description
2	RW	0x1	SM2DEV_FPCI_TIMEOUT_EN: FPCI timeout enable bit for Controller 0 = DISABLE 1 = ENABLE

22.16.8.41 XUSB_HOST_FPCI_TIMEOUT_0

FPCI_TIMEOUT Register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x000f0000 (0bxxxxxxxxxxxx11110000000000000000)

Bit	Reset	Description
19:0	0xf0000	SM2ALL_FPCI_TIMEOUT_THRESH: This field sets the timeout threshold value for the FPCI bus. It starts counting for each queue (ISO/NISO- RD/WR) with a pending request in the FPCI wrapper, the count resets when the requests are popped.

22.16.8.42 XUSB_HOST_TOM_0

Top of Memory Limit

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x3fff0fff (0bxx11111111111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3fff	LEG2ALL_TOM2: Top of Memory Limit 2.
11:0	0xfff	LEG2ALL_TOM1: Top of Memory Limit 1.

22.16.8.43 XUSB_HOST_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x1ac | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND: Number of pending initiator ISO PW responses

22.16.8.44 XUSB_HOST_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x1b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND: Number of pending initiator NISO PW responses

22.16.8.45 XUSB_HOST_INTR_STATUS_0

IPFS Interrupt Status

Offset: 0x1b4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	IP_INTR_STATUS: Status of IP (SATA/AZA) interrupt
1	X	MSI_INTR_STATUS: Status of MSI interrupt
0	X	IPFS_INTR_STATUS: Status of IPFS interrupt

22.16.8.46 XUSB_HOST_DFPCI_BEN_0

Downstream FPCI Byte Enables

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	0x0	EN_DFPCI_BEN: Enable bit for BEN. When set, the programmed BE is sent on the DFPCI bus
3:0	0x0	DFPCI_BYTE_ENABLE_N: Active low byte enables

22.16.8.47 XUSB_HOST_CLKGATE_HYSTERESIS_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000014 (0bxxxxxxxxxxxxxxxxxxxxxxxx00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of IPFS clock cycles to wait after clock gating criteria are met to disable IPFS/FPCI clocks

22.16.8.48 XUSB_HOST_XUSB_HOST_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

Note: *The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.*

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk field results in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to the legacy mode of operation (where the clock is on whenever the client clock is enabled).
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	XUSB_HOST_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	XUSB_HOST_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	XUSB_HOST_CCLK_OVERRIDE
17	0x0	XUSB_HOST_RCLK_OVERRIDE
16	0x0	XUSB_HOST_WCLK_OVERRIDE
3	DISABLE	XUSB_HOST_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	XUSB_HOST_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	XUSB_HOST_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	XUSB_HOST_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

22.16.8.49 XUSB_HOST_ORDERING_RULES_0

Offset: 0x1e0 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW: Modified Upstream MSIAW ordering. 0 = Tegra X1 MSIAW behavior 1 = Legacy (Tegra 3) MSIAW behavior
2	0x0	UPSTREAM_RESPAW: Modified RespAW ordering. 0 = Tegra X1 RespAW behavior 1 = Legacy (Tegra 3) RespAW behavior
1	0x0	UPSTREAM_RAW: Modified RAW ordering. 0 = Tegra X1 RAW behavior 1 = Legacy (Tegra 3) RAW behavior

22.16.8.50 XUSB_HOST_A2F_UFPCI_CFG0_0

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

22.16.8.51 XUSB_HOST_A2F_UFPCI_CFG1_0

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

22.16.9 XUSB Host PCI Config Registers

Base Address: XUSB_HOST_BASE + 0x8000

22.16.9.1 T_XUSB_CFG_0

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:16	FACh	R	T_XUSB_CFG_0_DEVICE_ID_UNIT: FA3h: DEVICE_ID_UNIT_XUSB (default)
15:0	10DEh	R	T_XUSB_CFG_0_VENDOR_ID: 10DEh: VENDOR_ID_NVIDIA (default)

22.16.9.2 T_XUSB_CFG_1

FPCI Configuration Register

Offset: 0x04 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	T_XUSB_CFG_1_DETECTED_PERR: The DETECTED_PERR bit indicates that the device has detected a parity error, even if parity error handing is disabled. (Bit 6 - T_CONFIG_FPCI_PERR_DISABLED) 0h: DETECTED_PERR_NOT_ACTIVE (default) 1h: DETECTED_PERR_ACTIVE 1h: DETECTED_PERR_CLEAR
30	0	R	T_XUSB_CFG_1_SIGNALED_SERR: The SIGNALED_SERR bit indicates that the device has asserted SERR#. 0h: SIGNALED_SERR_NOT_ACTIVE (default) 1h: SIGNALED_SERR_ACTIVE 1h: SIGNALED_SERR_CLEAR
29	0	R	T_XUSB_CFG_1_RECEIVED_MASTER: The RECEIVED_MASTER bit indicates that a master device's transaction (except for Special Cycle) was terminated with a master-abort. This means that no device on the PCI bus responded to the address of the mastered transaction. All master devices must implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_CLEAR
28	0	R	T_XUSB_CFG_1_RECEIVED_TARGET: The RECEIVED_TARGET bit indicates that a master device's transaction was terminated with a target-abort. All master devices must implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_CLEAR
27	0	R	T_XUSB_CFG_1_SIGNALED_TARGET: The SIGNALED_TARGET bit indicates that the device has terminated a transaction with target-abort. Devices that will never signal target-abort do not need to implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. 0h: SIGNALED_TARGET_NO_ABORT (default) 1h: SIGNALED_TARGET_ABORT 1h: SIGNALED_TARGET_CLEAR
26:25	0	R	T_XUSB_CFG_1_DEVSEL_TIMING: The DEVSEL_TIMING bits contain the timing of DEVSEL#. There are three allowable timings for assertion of DEVSEL#. These are encoded as 00b for fast, 01b for medium, and 10b for slow (11b is reserved). These bits are read only and must indicate the slowest time that a device asserts DEVSEL# for any bus command except Configuration Read and Configuration Write. FPCI positive decode device are required to respond with fast DEVSEL# (0-cycle). Only the subtractive FPCI function will respond with medium DEVSEL# to accept the cycle for the subtractive bus. 0h: DEVSEL_TIMING_FAST (default) 1h: DEVSEL_TIMING_MEDIUM 2h: DEVSEL_TIMING_SLOW
24	0	R	T_XUSB_CFG_1_MASTER_DATA_PERR: 0h: MASTER_DATA_PERR_NOT_ACTIVE (default) 1h: MASTER_DATA_PERR_ACTIVE 1h: MASTER_DATA_PERR_CLEAR
23	1h	R	T_XUSB_CFG_1_FAST_BACK2BACK: The FAST_BACK2BACK bit indicates that the device is capable of handling back-to-back transfers when the transactions are not to the same agent. This bit can be set to 1 if the device can accept these transactions, and must be set to 0 otherwise. 0h: FAST_BACK2BACK_INCAPABLE 1h: FAST_BACK2BACK_CAPABLE (default)
22	0	R	Reserved
21	1h	R	T_XUSB_CFG_1_66 MHZ: 0h: 66 MHZ_INCAPABLE 1h: 66 MHZ_CAPABLE (default)

Bits	Reset	R/W	Description
20	1h	R	T_XUSB_CFG_1_CAPLIST: The CAPLIST bit indicates that the device configuration space includes a capabilities list starting at the offset indicated by T_XUSB_DEV_CFG_13. 0h: CAPLIST_NOT_PRESENT 1h: CAPLIST_PRESENT (default)
19	None	R	T_XUSB_CFG_1_INTR_STATUS: The INTR_STATUS bit is read-only and reflects the state of the interrupt in the device/function. Only when the INTR_DISABLE bit in the command register is a 0 and this INTR_STATUS bit is a 1, will the device's/function's INTx# signal be asserted. Setting the INTR_DISABLE bit to 1 has no effect on the state of this bit. 0h: INTR_STATUS_0 1h: INTR_STATUS_1
18:11	0	R	Reserved
10	0	R/W	T_XUSB_CFG_1_INTR_DISABLE: The INTR_DISABLE bit indicates that it could disable the device/function from asserting INTx#. A value of 0 enables the assertion of INTx#, and a value of 1 disables the assertion of INTx# signal. The Device Status register is used to record status information for PCI bus related events. 0h: INTR_DISABLE_ON (default) 1h: INTR_DISABLE_OFF
9	0	R	T_XUSB_CFG_1_BACK2BACK: 0h: BACK2BACK_DISABLED (default) 1h: BACK2BACK_ENABLED
8	0	R	T_XUSB_CFG_1_SERR: 0h: SERR_DISABLED (default) 1h: SERR_ENABLED
7	0	R	T_XUSB_CFG_1_STEP: 0h: STEP_DISABLED (default) 1h: STEP_ENABLED
6	0	R	T_XUSB_CFG_1_PERR: 0h: PERR_DISABLED (default) 1h: PERR_ENABLED
5	0	R	T_XUSB_CFG_1_PALETTE_SNOOP: The PALETTE_SNOOP bit indicates that VGA compatible devices should snoop their palette registers. When this bit is set, special palette snooping behavior is enabled (i.e., device must not respond). When the bit is reset, the device should treat palette accesses like all other accesses. VGA compatible devices should implement this bit. PALETTE_SNOOP is writable. 0h: PALETTE_SNOOP_DISABLED (default) 1h: PALETTE_SNOOP_ENABLED
4	0	R	T_XUSB_CFG_1_WRITE_AND_INVAL: The WRITE_AND_INVAL bit indicates that the device can use the Memory Write and Invalidate command when the transfer is aligned and 16 bytes and the contents of the Cache Line Size Register is 4 DWORDS. When this bit is 1, masters may generate the command. When it is 0, Memory Write must be used instead. State after RST# is 0. This bit must be implemented by master devices that can generate the Memory Write and Invalidate command. 0h: WRITE_AND_INVAL_DISABLED (default) 1h: WRITE_AND_INVAL_ENABLED
3	0	R	T_XUSB_CFG_1_SPECIAL_CYCLE: 0h: SPECIAL_CYCLE_DISABLED (default) 1h: SPECIAL_CYCLE_ENABLED
2	0	R/W	T_XUSB_CFG_1_BUS_MASTER: The BUS_MASTER bit indicates that the device can act as a master on the PCI bus. A value of 0 disables the device from generating PCI accesses. A value of 1 allows the device to behave as a bus master. BUS_MASTER is writable. 0h: BUS_MASTER_DISABLED (default) 1h: BUS_MASTER_ENABLED
1	0	R/W	T_XUSB_CFG_1_MEMORY_SPACE: The MEMORY_SPACE bit indicates that the device will respond to memory space accesses. A value of 0 disables the device response. A value of 1 allows the device to respond to Memory space accesses. MEMORY_SPACE is writable. 0h: MEMORY_SPACE_DISABLED (default) 1h: MEMORY_SPACE_ENABLED

Bits	Reset	R/W	Description
0	0	R/W	T_XUSB_CFG_1_IO_SPACE: The IO_SPACE bit indicates that the device will respond to I/O space accesses. A value of 0 disables the device response. A value of 1 allows the device to respond to I/O space accesses. IO_SPACE is writable. 0h: IO_SPACE_DISABLED (default) 1h: IO_SPACE_ENABLED

22.16.9.3 T_XUSB_CFG_2

PCI Revision ID and Class Code Register

Offset: 0x08 | Read/Write: R

Bits	Reset	R/W	Description
31:24	Ch	R	T_XUSB_CFG_2_BASE_CLASS: The CLASS_CODE bits identify the generic function of the device and (in some cases) a specific register-level programming interface. The register is broken into three byte-size fields. The upper byte (at offset 0BH) is a base class code which broadly classifies the type of function the device performs. The middle-byte (at offset 0BH) is a sub-class code which identifies more specifically the function of the device. The lower byte (at offset 09H) identifies a specific register-level programming interface (if any) so that device independent software can interact with the device. The Class Code and Revision ID are defined by parameters per block. Ch: BASE_CLASS_SBC (default)
23:16	3h	R	T_XUSB_CFG_2_SUB_CLASS: 3h: SUB_CLASS_XUSB (default)
15:8	FEh	R	T_XUSB_CFG_2_PROG_IF: FEh: PROG_IF_XHCI (default)
7:0	A1h	R	T_XUSB_CFG_2_REVISION_ID: The REVISION_ID bits specify a device specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. This field should be viewed as a vendor defined extension to the DEVICE_ID. A1h: REVISION_ID_VAL (default) A1h: REVISION_ID_A01

22.16.9.4 T_XUSB_CFG_3

PCI Configuration Register

Offset: 0x0C | Read/Write: R

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23	0	R	T_XUSB_CFG_3_HEADER_TYPE_FUNC: 0h: HEADER_TYPE_FUNC_SINGLE (default) 1h: HEADER_TYPE_FUNC_MULTI
22:16	0	R	T_XUSB_CFG_3_HEADER_TYPE_DEVICE: The HEADER_TYPE bits identify the layout of the bytes 10h through 3Fh in configuration space and also whether or not the device contains multiple functions. Bit 7 in this register is used to identify a multi-function device. If the bit is 0, then the device is single function. If the bit is 1, then the device has multiple functions. Bits 6 through 0 specify the layout of bytes 10h through 3Fh. The LATENCY_TIMER and HEADER_TYPE are defined by parameters per block. 0h: HEADER_TYPE_DEVICE_NON_BRIDGE (default) 1h: HEADER_TYPE_DEVICE_P2P_BRIDGE
15:11	0	R	T_XUSB_CFG_3_LATENCY_TIMER: The LATENCY_TIMER bits contain, in units of PCI bus clocks, the value of the Latency Timer for this PCI bus master. This register must be implemented as writable by any master that can burst more than two data phases. This register may be implemented as read-only for devices that burst two or fewer data phases, but the hardwired value must be limited to 16 or less. A typical implementation would be to build the five high-order bits (leaving the bottom three as read-only), resulting in a timer granularity of eight clocks. At reset, the register should be set to 0 (if programmable). LATENCY_TIMER bits are writable. 0h: LATENCY_TIMER_0_CLOCKS (default) 1h: LATENCY_TIMER_8_CLOCKS 1Eh: LATENCY_TIMER_240_CLOCKS 1Fh: LATENCY_TIMER_248_CLOCKS

Bits	Reset	R/W	Description
10:8	0	R	Reserved
7:0	0	R	T_XUSB_CFG_3_CACHE_LINE_SIZE: 0h: CACHE_LINE_SIZE_0 (default) 20h: CACHE_LINE_SIZE_32 40h: CACHE_LINE_SIZE_64

22.16.9.5 T_XUSB_CFG_4

PCI Configuration Register

Offset: 0x10 | Read/Write: R/W

Bits	Reset	R/W	Description
31:15	0	R/W	T_XUSB_CFG_4_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device required by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all don't-care address bits, effectively specifying the address space required. 0h: BASE_ADDRESS_DEFAULT (default)
14:4	0	R	T_XUSB_CFG_4_BAR_SIZE_32KB: 0h: BAR_SIZE_32KB_RSVD (default)
3	1h	R	T_XUSB_CFG_4_PREFETCHABLE: 0h: PREFETCHABLE_NOT 1h: PREFETCHABLE_MERGABLE (default)
2:1	2h	R	T_XUSB_CFG_4_ADDRESS_TYPE: The ADDRESS_TYPE bits contain the type of the Base Address. It can be 32 bits, 20 bits, or 64 bits wide. 0h: ADDRESS_TYPE_32_BIT 2h: ADDRESS_TYPE_64_BIT (default)
0	0	R	T_XUSB_CFG_4_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (default) 1h: SPACE_TYPE_IO

22.16.9.6 T_XUSB_CFG_5

Offset: 0x14 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_CFG_5_BASE_ADDRESSHI: 0h: BASE_ADDRESSHI_DEFAULT (default)

22.16.9.7 T_XUSB_CFG_6

Offset: 0x18-0x28 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_6_RSVD: 0h: RSVD_00 (default)

22.16.9.8 T_XUSB_CFG_11

PCI Configuration Register

The SUBSYSTEM_VENDOR_ID bits and SUBSYSTEM_ID bits are used to uniquely identify the add-in board or subsystem where the device resides. When the device is on the motherboard, there is no serial ROM and the registers both initialize to NONE. The motherboard BIOS must set the values of the Subsystem ID and Subsystem Vendor ID by writing the proper values to the SUBSYSTEM_VENDOR_ID and SUBSYSTEM_ID bits in the PCI_T_16 register (NOT PCI_T_11).

Offset: 0x2c | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_CFG_11_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R	T_XUSB_CFG_11_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

22.16.9.9 T_XUSB_CFG_12

PCI Configuration Register

Offset: 0x30 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_12_RESERVED: 0h: RESERVED_0 (default)

22.16.9.10 T_XUSB_CFG_13

PCI Capability Pointer Register

Offset: 0x34 | Read/Write: R

Bits	Reset	R/W	Description
31:8	0	R	Reserved
7:0	44h	R	T_XUSB_CFG_13_CAP_PTR: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. This always points to 0x44 where at least the PCI-PM registers are expected to reside. 44h: CAP_PTR_PMCAP C0h: CAP_PTR_MSI 70h: CAP_PTR_MSIX DCh: CAP_PTR_MMAP 44h: CAP_PTR_DEFAULT (default)

22.16.9.11 T_XUSB_CFG_14

PCI Configuration Register

Offset: 0x38 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_14_RESERVED: 0h: RESERVED_0 (default)

22.16.9.12 T_XUSB_CFG_15

PCI Configuration Register

Offset: 0x3c | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_CFG_15_MAX_LAT: The MAX_LAT bits contain the maximum time the device requires to gain access to the CPI bus. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microseconds. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MAX_LAT is nonzero. The INTR_PIN, MIN_GNT, and MAX_LAT are configurable per block. For a 64-byte buffer with two 32-byte sections, the maximum tolerable latency for the XHCI once a large XUSB ISO transaction has begun is about 23 µs. The latency timer is hardwired to 20 µs. This value only applies in External PCI operation. 0h: MAX_LAT_NO_REQUIREMENTS (default) 14h: MAX_LAT_5US 50h: MAX_LAT_20US
23:16	0	R	T_XUSB_CFG_15_MIN_GNT: The MIN_GNT bits contain the length of the burst period a device needs assuming a clock rate of 33 MHz. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MIN_GNT is nonzero. 0h: MIN_GNT_NO_REQUIREMENTS (default) 1h: MIN_GNT_240NS
15:8	1h	R	T_XUSB_CFG_15_INTR_PIN: The INTR_PIN bits contain the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that do not use an interrupt pin must put a 0 in this register. 0h: INTR_PIN_NONE 1h: INTR_PIN_INTA (default) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	0	R/W	T_XUSB_CFG_15_INTR_LINE: The INTR_LINE bits contain the interrupt routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this register as it initializes and configures the system. The value in this register tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is initialized to 0xff (no connection) at reset. Some PCI BIOS' cannot handle aliased INTR_LINEs. Some PCI BIOS' cannot handle INTR_LINE initialized to 0xff. 0h: INTR_LINE_IRQ0 (default) 1h: INTR_LINE_IRQ1 Fh: INTR_LINE_IRQ15 FFh: INTR_LINE_UNKNOWN

22.16.9.13 T_XUSB_CFG_16

Backdoor register write for updating subsystem ID/vendor ID.

Offset: 0x40 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_CFG_16_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R/W	T_XUSB_CFG_16_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

22.16.9.14 T_XUSB_CFG_17

Offset: 0x44 | Read/Write: R

Bits	Reset	R/W	Description
31	1h	R	T_XUSB_CFG_17_D3CPME_SUPPORT: 1h: D3CPME_SUPPORT_YES 0h: D3CPME_SUPPORT_NO
30	1h	R	T_XUSB_CFG_17_D3HPME_SUPPORT: 1h: D3HPME_SUPPORT_YES 0h: D3HPME_SUPPORT_NO
29	0	R	T_XUSB_CFG_17_D2PME_SUPPORT: 1h: D2PME_SUPPORT_YES 0h: D2PME_SUPPORT_NO
28	0	R	T_XUSB_CFG_17_D1PME_SUPPORT: 1h: D1PME_SUPPORT_YES 0h: D1PME_SUPPORT_NO
27	1h	R	T_XUSB_CFG_17_D0PME_SUPPORT: 1h: D0PME_SUPPORT_YES 0h: D0PME_SUPPORT_NO
26	0	R	T_XUSB_CFG_17_D2_SUPPORT: 0h: D2_SUPPORT_NO 1h: D2_SUPPORT_YES
25	0	R	T_XUSB_CFG_17_D1_SUPPORT: 0h: D1_SUPPORT_NO 1h: D1_SUPPORT_YES
24:22	0	R	T_XUSB_CFG_17_AUXCUR: 0h: AUXCUR_SELF 1h: AUXCUR_55MA 2h: AUXCUR_100MA 3h: AUXCUR_160MA 4h: AUXCUR_220MA 5h: AUXCUR_270MA 6h: AUXCUR_320MA 7h: AUXCUR_375MA
21	0	R	T_XUSB_CFG_17_DSI: 0h: DSI_NONE 1h: DSI_NEEDED
20	0	R	T_XUSB_CFG_17_RSVD: 0h: RSVD_0
19	0	R	T_XUSB_CFG_17_PMECLK: 0h: PMECLK_NOT_REQUIRED 1h: PMECLK_REQUIRED
18:16	3h	R	T_XUSB_CFG_17_VER: 3h: VER_1P2
15:8	C0h	R	T_XUSB_CFG_17_NEXT_PTR: C0h: NEXT_PTR_MSI 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 0h: NEXT_PTR_NULL C0h: NEXT_PTR_DEFAULT (default)
7:0	1h	R	T_XUSB_CFG_17_CAP: 1h: CAP_PCIPM

22.16.9.15 T_XUSB_CFG_18_PMCSR

Offset: 0x48 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23:16	0	R	T_XUSB_CFG_18_PMCSR_BSE_RSVD: 0h: BSE_RSVD_00

Bits	Reset	R/W	Description
15	0	RW1C	T_XUSB_CFG_18_PMCSR_PMESTATUS: 0h: PMESTATUS_NOT_PENDING (default) 1h: PMESTATUS_PENDING 1h: PMESTATUS_CLEAR
14:13	0	R	T_XUSB_CFG_18_PMCSR_DSCALE: 0h: DSCALE_INIT
12:9	0	R	T_XUSB_CFG_18_PMCSR_DSEL: 0h: DSEL_INIT
8	0	R/W	T_XUSB_CFG_18_PMCSR_PME: 1h: PME_ENABLE 0h: PME_DISABLE (default)
7:4	0	R	T_XUSB_CFG_18_PMCSR_RSVD1: 0h: RSVD1_00
3	1h	R	T_XUSB_CFG_18_PMCSR_NSR: 1h: NSR_NORESET 0h: NSR_RESET
2	0	R	T_XUSB_CFG_18_PMCSR_RSVD0: 0h: RSVD0_0
1:0	0	R/W	T_XUSB_CFG_18_PMCSR_PWRSTATE: 0h: PWRSTATE_D0 (default) 1h: PWRSTATE_D1 2h: PWRSTATE_D2 3h: PWRSTATE_D3H

22.16.9.16 T_XUSB_CFG_24

XUSB XHCI Configuration Control

Offset: 0x60 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:14	0	R	T_XUSB_CFG_24_RSVDP: 0h: RSVDP_VALUE
13:8	20h	R/W	T_XUSB_CFG_24_FLADJ: Frame Length Adjustment Register. This register is in the auxiliary power well. This feature is used to adjust any offset from the clock source that generates the clock that drives the SOF counter. When a new value is written into these six bits, the length of the frame is adjusted. Its initial programmed value is system dependent based on the accuracy of hardware USB clock and is initialized by system BIOS. This register should only be modified when the HChalted bit in the USBSTS register is a one. Changing value of this register while the host controller is operating yields undefined results. It should not be reprogrammed by USB system software unless the default or BIOS programmed values are incorrect, or the system is restoring the register while returning from a suspended state. 20h: FLADJ_VALUE (default)
7:0	30h	R	T_XUSB_CFG_24_SBRN: Serial Bus Release Number Register. This register contains the release of the Universal Serial Bus Specification with which this Universal Serial Bus Host Controller module is compliant. 30h: SBRN_VALUE

22.16.9.17 T_XUSB_MSI_CTRL

MSI Message Control and Capability Register B0h

MSI_CTRL is the main control register for MSI support.

Offset: 0xc0 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	Reserved

Bits	Reset	R/W	Description
24	0	R	T_XUSB_MSI_CTRL_VECTOR_MASK_CAP: The VECTOR_MASK_CAP field indicates whether or not the controller supports MSI-per-vector masking. 0h: VECTOR_MASK_CAP_DIS 1h: VECTOR_MASK_CAP_EN 0h: VECTOR_MASK_CAP_DEFAULT (default)
23	1h	R	T_XUSB_MSI_CTRL_64_ADDR_CAP: The 64_ADDR_CAP field indicates whether or not the controller is capable of generating a 64-bit message address. A value of 1 means the controller is capable of generating a 64-bit message address. 0h: 64_ADDR_CAP_DIS 1h: 64_ADDR_CAP_EN 1h: 64_ADDR_CAP_DEFAULT (default)
22:20	0	R/W	T_XUSB_MSI_CTRL_MULT_MSG_ENABLE: System software writes to this field to indicate the number of allocated vectors (less than or equal to the number of vectors requested). The number of vectors is aligned as a power of two. When MSI is enabled, the controller will be allocated at least one vector. 0h: MULT_MSG_ENABLE_1 1h: MULT_MSG_ENABLE_2 2h: MULT_MSG_ENABLE_4 3h: MULT_MSG_ENABLE_8 4h: MULT_MSG_ENABLE_16 5h: MULT_MSG_ENABLE_32 0h: MULT_MSG_ENABLE_DEFAULT (default)
19:17	0	R	T_XUSB_MSI_CTRL_MULT_MSG_CAP: System software reads this field to determine the number of requested vectors. The number of requested vectors must be aligned to a power of two. Values of 6 and 7 in this field are reserved. 0h: MULT_MSG_CAP_1 1h: MULT_MSG_CAP_2 2h: MULT_MSG_CAP_4 3h: MULT_MSG_CAP_8 4h: MULT_MSG_CAP_16 5h: MULT_MSG_CAP_32 0h: MULT_MSG_CAP_DEFAULT (default)
16	0	R/W	T_XUSB_MSI_CTRL_MSI_ENABLE: The MSI_ENABLE field enables the MSI capability. If MSI_ENABLE is written to a 1, the controller is permitted to use MSI to request service and is prohibited from using the legacy interrupt. System configuration software sets this bit to enable MSI. A device driver is prohibited from writing this bit to mask the controller's service request. If this bit is written to a 0, the controller is prohibited from using MSI to request service. 0h: MSI_ENABLE_OFF 1h: MSI_ENABLE_ON 0h: MSI_ENABLE_DEFAULT (default)
15:8	E0h	R	T_XUSB_MSI_CTRL_NEXT_PTR: The NEXT_PTR field identifies the next item in the capabilities list. It is a read-only field. 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 44h: NEXT_PTR_PMCAP 0h: NEXT_PTR_NULL E0h: NEXT_PTR_MAILBOX E0h: NEXT_PTR_DEFAULT (default)
7:0	5h	R	T_XUSB_MSI_CTRL_CAP_ID: The CAP_ID field identifies this capability block as the MSI capability block. This is read-only as 0x5. 5h: CAP_ID_MSI (default)

22.16.9.18 T_XUSB_MSI_ADDR1

MSI Message Address Register 84h

MSI_ADDR1 specifies the lower 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc4 | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R/W	T_XUSB_MSI_ADDR1_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the Dword-aligned address for the MSI memory write transaction. 0h: MSG_ADDR_DEFAULT (default)
1:0	0	R	Reserved

22.16.9.19 T_XUSB_MSI_ADDR2

MSI Message Upper Address Register 88h

MSI_ADDR2 specifies the upper 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc8 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_MSI_ADDR2_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the upper 32 bits of the address for the MSI memory write transaction. The contents of this register only apply when T_XUSB_CFG_MSI_CTRL_64_ADDR_CAP bit is set. 0h: MSG_ADDR_DEFAULT (default)

22.16.9.20 T_XUSB_MSI_DATA

MSI Message Data Register 8Ch

The MSI_DATA register contains the system-specified message.

Offset: 0xcc | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	0	R/W	T_XUSB_MSI_DATA_MSG_DATA: System-specified message When MSI is enabled, the message data is driven onto the lower 16 bits of the MSI memory write. The MULT_MSG_ENABLE field in configuration register 80h specifies the number of low-order message data bits that the XHCI is permitted to modify to generate its system software allocated vectors. 0h: MSG_DATA_DEFAULT (default)

22.16.9.21 T_XUSB_MSI_MASK

MSI Message Data Register 60h

The MSI_MASK register enables software to mask or defer message generation on a per-vector basis.

Offset: 0xD0 | Read/Write: R/W

Bits	Reset	R/W	Description
31:1	0	R/W	Reserved
0	0	R/W	T_XUSB_MSI_MASK_BIT: For each mask bit that is set, the controller is prohibited from generating the associated message. bit 0 corresponds to MSI vector 0 bit 1 corresponds to MSI vector 1 bit 2 corresponds to MSI vector 2 bit 3 corresponds to MSI vector 3 bit 4 corresponds to MSI vector 4 bit 5 corresponds to MSI vector 5 bit 6 corresponds to MSI vector 6 bit 7 corresponds to MSI vector 7, etc. 0h: MSI_MASK_BIT_DEFAULT (default)

22.16.9.22 T_XUSB_MSI_PEND

MSI Pending Bits Register 64h

The MSI_PEND register shows which MSI vectors have pending interrupts.

Offset: 0xD4 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	0	R	T_XUSB_MSI_PEND_BIT: For each Pending bit that is set, XUSB has a pending associated message. bit 0 corresponds to MSI vector 0 bit 1 corresponds to MSI vector 1 bit 2 corresponds to MSI vector 2 bit 3 corresponds to MSI vector 3 bit 4 corresponds to MSI vector 4 bit 5 corresponds to MSI vector 5 bit 6 corresponds to MSI vector 6 bit 7 corresponds to MSI vector 7, etc. 0h: MSI_PEND_BIT_DEFAULT (default)

22.16.10 XUSB Mailbox Registers

22.16.10.1 T_XUSB_CFG_ARU_MAILBOX_CAP

Offset: 0xe0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:24	0	R/W	T_XUSB_CFG_ARU_MAILBOX_CAP_RSVD: 0h: MAILBOX_CAP_RSVD_INIT (default)
23:16	14h	RW	T_XUSB_CFG_ARU_MAILBOX_CAP_LENGTH: 14h: MAILBOX_CAP_LENGTH_INIT (default)
15:8	0	RW	T_XUSB_CFG_ARU_MAILBOX_CAP_NEXTPTR 0h: MAILBOX_CAP_NEXTPTR_NULL 0h: MAILBOX_CAP_NEXTPTR_INIT (default)
7:0	9h	RW	T_XUSB_CFG_ARU_MAILBOX_CAP_ID: 9h: MAILBOX_CAP_ID_INIT (default)

22.16.10.2 T_XUSB_CFG_ARU_MAILBOX_CMD

Offset: 0xe4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31	0	R/W	T_XUSB_CFG_ARU_MAILBOX_CMD_INT_EN: 0h: MAILBOX_CMD_INT_EN_INIT (default)
30	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_XHCI: 0h: MAILBOX_CMD_DEST_XHCI_INIT (default)
29	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_SMI: 0h: MAILBOX_CMD_DEST_SMI_INIT (default)
28	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_PME: 0h: MAILBOX_CMD_DEST_PME_INIT (default)
27	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_DEST_FALCON: 0h: MAILBOX_CMD_DEST_FALCON_INIT (default)
26:0	0	RW	T_XUSB_CFG_ARU_MAILBOX_CMD_RSVD: 0h: MAILBOX_CMD_RSVD_INIT (default)

22.16.10.3 T_XUSB_CFG_ARU_MAILBOX_DATA_IN

Offset: 0xe8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_CFG_ARU_MAILBOX_DATA_IN_MSG: 0h: MAILBOX_DATA_IN_MSG_INIT (default)

22.16.10.4 T_XUSB_CFG_ARU_MAILBOX_DATA_OUT

Offset: 0xec | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_CFG_ARU_MAILBOX_DATA_OUT_MSG: 0h: MAILBOX_DATA_OUT_MSG_INIT (default)

22.16.10.5 T_XUSB_CFG_ARU_MAILBOX_OWNER

Offset: 0xf0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b00000000000000000000000000000000)

Bits	Reset	R/W	Description
31:8	0	R/W	T_XUSB_CFG_ARU_MAILBOX_ OWNER_RSVD: 0h: MAILBOX_OWNER_RSVD_INIT (default)
7:0	0	RW	T_XUSB_CFG_ARU_MAILBOX_ OWNER_ID: 0h: MAILBOX_OWNER_ID_INIT (default)

22.16.11 XUSB Host XHCI Registers

Base Address: XUSB_HOST_BASE

22.16.11.1 T_XUSB_XHCI_CAP_REG0

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_REG0_HCIVERSION: 100h: HCIVERSION_INIT
15:8	Unknown	R	T_XUSB_XHCI_CAP_REG0_RSVD: 0h: RSVD_00
7:0	Unknown	R	T_XUSB_XHCI_CAP_REG0_CAPLENGTH: 20h: CAPLENGTH_INIT

22.16.11.2 T_XUSB_XHCI_CAP_HCSPARAMS1

Offset: 0x04 | Read/Write: R

Bits	Reset	R/W	Description
31:24	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXPORTS: 10h: MAXPORTS_INIT
23	0	R	Reserved
22:19	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_RSVD0: 0h: RSVD0_00
18:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXINTRS: 1h: MAXINTRS_INIT
7:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXSLOTS: FFh: MAXSLOTS_INIT

22.16.11.3 T_XUSB_XHCI_CAP_HCSPARAMS2

Offset: 0x08 | Read/Write: R

Bits	Reset	R/W	Description
31:27	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_MAXSPBLO: 0h: MAXSPBLO_INIT
26	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_SPR: 1h: SPR_TRUE 0h: SPR_FALSE
25:21	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_MAXSPBHI: 0h: MAXSPBHI_INIT
20:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_RSVD: 0h: RSVD_00
7:4	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_ERST_MAX: Fh: ERST_MAX_INIT
3:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_IST: 8h: IST_INIT

22.16.11.4 T_XUSB_XHCI_CAP_HCSPARAMS3

Offset: 0x0c | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_U2LAT: 40h: U2LAT_INIT
15:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_RSVD: 0h: RSVD_00
7:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_U1LAT: 2h: U1LAT_INIT

22.16.11.5 T_XUSB_XHCI_CAP_HCCPARAMS

Offset: 0x10 | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_XECP: 180h: XECP_USBLEGSUP 184h: XECP_SUPPROT_USB3 187h: XECP_SUPPROT_USB2
15:12	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_MAXPSASIZE: Fh: MAXPSASIZE_INIT
11:9	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_RSVD: 0h: RSVD_00
8	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PAE: 1h: PAE_TRUE 0h: PAE_FALSE
7	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_NSS: 1h: NSS_TRUE 0h: NSS_FALSE
6	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_LTC: 1h: LTC_TRUE 0h: LTC_FALSE
5	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_LHRC: 1h: LHRC_TRUE 0h: LHRC_FALSE
4	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PIND: 1h: PIND_TRUE 0h: PIND_FALSE
3	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PPC: 1h: PPC_TRUE 0h: PPC_FALSE

Bits	Reset	R/W	Description
2	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_CSZ: 1h: CSZ_64B 0h: CSZ_32B
1	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_BNC: 1h: BNC_TRUE 0h: BNC_FALSE
0	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_AC64: 1h: AC64_TRUE 0h: AC64_FALSE

22.16.11.6 T_XUSB_XHCI_CAP_DBOFF

Offset: 0x14 | Read/Write: R

Bits	Reset	R/W	Description
31:2	Unknown	R	T_XUSB_XHCI_CAP_DBOFF_OFFSET: 300h: OFFSET_INIT
1:0	Unknown	R	T_XUSB_XHCI_CAP_DBOFF_RSVD: 0h: RSVD_00

22.16.11.7 T_XUSB_XHCI_CAP_RTSOFF

Offset: 0x18 | Read/Write: R

Bits	Reset	R/W	Description
31:5	Unknown	R	T_XUSB_XHCI_CAP_RTSOFF_OFFSET: 40h: OFFSET_INIT
4:0	Unknown	R	T_XUSB_XHCI_CAP_RTSOFF_RSVD: 0h: RSVD_00

22.16.11.8 T_XUSB_XHCI_CAP_RSVD0

Offset: 0x1c | Read/Write: R

Bits	Reset	R/W	Description
31:0	Unknown	R	T_XUSB_XHCI_CAP_RSVD0_F: 0h: F_00

22.16.11.9 T_XUSB_XHCI_OP_USBCMD

Offset: 0x20 | Read/Write: R/W

Bits	Reset	R/W	Description
31:12	0	R	T_XUSB_XHCI_OP_USBCMD_RSVD1: 0h: RSVD1_00 (default)
11	0	R/W	T_XUSB_XHCI_OP_USBCMD_EU3S: 0h: EU3S_DISABLE (default) 1h: EU3S_ENABLE
10	0	R/W	T_XUSB_XHCI_OP_USBCMD_EWE: 0h: EWE_DISABLE (default) 1h: EWE_ENABLE
9	0	RW1C	T_XUSB_XHCI_OP_USBCMD_CRS: 0h: CRS_INIT (default) 1h: CRS_START 0h: CRS_NOOP
8	0	RW1C	T_XUSB_XHCI_OP_USBCMD_CSS: 0h: CSS_INIT (default) 1h: CSS_START 0h: CSS_NOOP

Bits	Reset	R/W	Description
7	0	R/W	T_XUSB_XHCI_OP_USBCMD_LHCRST: 0h: LHCRST_NOT_PENDING (default) 1h: LHCRST_PENDING 1h: LHCRST_SET
6:4	0	R	T_XUSB_XHCI_OP_USBCMD_RSVD0: 0h: RSVD0_00 (default)
3	0	R/W	T_XUSB_XHCI_OP_USBCMD_HSEE: 0h: HSEE_DISABLE (default) 1h: HSEE_ENABLE
2	0	R/W	T_XUSB_XHCI_OP_USBCMD_INTE: 0h: INTE_DISABLE (default) 1h: INTE_ENABLE
1	0	R/W	T_XUSB_XHCI_OP_USBCMD_HCRST: 0h: HCRST_NOT_PENDING (default) 1h: HCRST_PENDING 1h: HCRST_SET
0	0	R/W	T_XUSB_XHCI_OP_USBCMD_RS: 0h: RS_STOP (default) 1h: RS_RUN

22.16.11.10 T_XUSB_XHCI_OP_USBSTS

Offset: 0x24 | Read/Write: R/W

Bits	Reset	R/W	Description
31:13	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD2: 0h: RSVD2_00 (default)
12	0	R	T_XUSB_XHCI_OP_USBSTS_HCE: 0h: HCE_NO_ERROR (default) 1h: HCE_ERROR
11	1	R	T_XUSB_XHCI_OP_USBSTS_CNR: 1h: CNR_NOT_READY (default) 0h: CNR_READY
10	0	RW1C	T_XUSB_XHCI_OP_USBSTS_SRE: 0h: SRE_NOT_PENDING (default) 1h: SRE_PENDING 1h: SRE_CLEAR
9	0	R	T_XUSB_XHCI_OP_USBSTS_RSS: 0h: RSS_NOT_PENDING (default) 1h: RSS_PENDING
8	0	R	T_XUSB_XHCI_OP_USBSTS_SSS: 0h: SSS_NOT_PENDING (default) 1h: SSS_PENDING
7:5	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD1: 0h: RSVD1_00 (default)
4	0	RW1C	T_XUSB_XHCI_OP_USBSTS_PCD: 0h: PCD_NOT_PENDING (default) 1h: PCD_PENDING 1h: PCD_CLEAR
3	0	RW1C	T_XUSB_XHCI_OP_USBSTS_EINT: 0h: EINT_NOT_PENDING (default) 1h: EINT_PENDING 1h: EINT_CLEAR
2	0	RW1C	T_XUSB_XHCI_OP_USBSTS_HSE: 0h: HSE_NOT_PENDING (default) 1h: HSE_PENDING 1h: HSE_CLEAR
1	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD0: 0h: RSVD0_0 (default)
0	1	R	T_XUSB_XHCI_OP_USBSTS_HCH: 1h: HCH_HALTED (default) 0h: HCH_RUNNING

22.16.11.11 T_XUSB_XHCI_OP_PGSZ

Offset: 0x28 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_OP_PGSZ_RSVD0: 0h: RSVD0_00 (default)
15:0	1	R	T_XUSB_XHCI_OP_PGSZ_PAGESIZE: 1h: PAGESIZE_4K (default)

22.16.11.12 T_XUSB_XHCI_OP_DNCTRL

Offset: 0x34 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N15: 0h: N15_DISABLE (default) 1h: N15_ENABLE
14	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N14: 0h: N14_DISABLE (default) 1h: N14_ENABLE
13	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N13: 0h: N13_DISABLE (default) 1h: N13_ENABLE
12	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N12: 0h: N12_DISABLE (default) 1h: N12_ENABLE
11	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N11: 0h: N11_DISABLE (default) 1h: N11_ENABLE
10	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N10: 0h: N10_DISABLE (default) 1h: N10_ENABLE
9	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N9: 0h: N9_DISABLE (default) 1h: N9_ENABLE
8	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N8: 0h: N8_DISABLE (default) 1h: N8_ENABLE
7	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N7: 0h: N7_DISABLE (default) 1h: N7_ENABLE
6	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N6: 0h: N6_DISABLE (default) 1h: N6_ENABLE
5	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N5: 0h: N5_DISABLE (default) 1h: N5_ENABLE
4	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N4: 0h: N4_DISABLE (default) 1h: N4_ENABLE
3	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N3: 0h: N3_DISABLE (default) 1h: N3_ENABLE
2	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N2: 0h: N2_DISABLE (default) 1h: N2_ENABLE
1	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N1: 0h: N1_DISABLE (default) 1h: N1_ENABLE

Bits	Reset	R/W	Description
0	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N0: 0h: N0_DISABLE (default) 1h: N0_ENABLE

22.16.11.13 T_XUSB_XHCI_OP_CRCR0

Offset: 0x38 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	W	T_XUSB_XHCI_OP_CRCR0_CRPLO: 0h: CRPLO_INIT (default)
5:4	0	R	T_XUSB_XHCI_OP_CRCR0_RSVD0: 0h: RSVD0_00 (default)
3	0	R	T_XUSB_XHCI_OP_CRCR0_CRR: 0h: CRR_STOPPED (default) 1h: CRR_RUNNING
2	0	RW1C	T_XUSB_XHCI_OP_CRCR0_CA: 0h: CA_INIT (default) 1h: CA_ABORT
1	0	RW1C	T_XUSB_XHCI_OP_CRCR0_CS: 0h: CS_INIT (default) 1h: CS_STOP
0	0	RW1C	T_XUSB_XHCI_OP_CRCR0_RCS: 0h: RCS_0 (default) 1h: RCS_1

22.16.11.14 T_XUSB_XHCI_OP_CRCR1

Offset: 0x3c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	W	T_XUSB_XHCI_OP_CRCR1_CRPHI: 0h: CRPHI_INIT (default)

22.16.11.15 T_XUSB_XHCI_OP_DCBAAP0

Offset: 0x50 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	R/W	T_XUSB_XHCI_OP_DCBAAP0_DCBAAPLO: 0h: DCBAAPLO_INIT (default)
5:0	0	R	T_XUSB_XHCI_OP_DCBAAP0_RSVD0: 0h: RSVD0_00 (default)

22.16.11.16 T_XUSB_XHCI_OP_DCBAAP1

Offset: 0x54 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_OP_DCBAAP1_DCBAAPHI: 0h: DCBAAPHI_INIT (default)

22.16.11.17 T_XUSB_XHCI_OP_CONFIG

Offset: 0x58 | Read/Write: R/W

Bits	Reset	R/W	Description
31:8	0	R	T_XUSB_XHCI_OP_CONFIG_RSVD0: 0h: RSVD0_00 (default)

Bits	Reset	R/W	Description
7:0	0	R/W	T_XUSB_XHCI_OP_CONFIG_MAXSLOTSEN: 0h: MAXSLOTSEN_INIT (default)

22.16.11.18 T_XUSB_XHCI_OP_PORTSC

Offset: 0x420 – 0x510 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	T_XUSB_XHCI_OP_PORTSC_WPR: 0h: WPR_NOT_PENDING (default) 1h: WPR_PENDING 1h: WPR_SET
30	0	R	T_XUSB_XHCI_OP_PORTSC_DR: 0h: DR_FALSE (default) 1h: DR_TRUE
29:28	0	R	T_XUSB_XHCI_OP_PORTSC_RSVD2: 0h: RSVD2_00 (default)
27	0	R/W	T_XUSB_XHCI_OP_PORTSC_WOE: 0h: WOE_DISABLED (default) 1h: WOE_ENABLED
26	0	R/W	T_XUSB_XHCI_OP_PORTSC_WDE: 0h: WDE_DISABLED (default) 1h: WDE_ENABLED
25	0	R/W	T_XUSB_XHCI_OP_PORTSC_WCE: 0h: WCE_DISABLED (default) 1h: WCE_ENABLED
24	0	R	T_XUSB_XHCI_OP_PORTSC_CAS: 0h: CAS_INIT (default)
23	0	RW1C	T_XUSB_XHCI_OP_PORTSC_CEC: 0h: CEC_NOT_PENDING (default) 1h: CEC_PENDING 1h: CEC_CLEAR
22	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PLC: 0h: PLC_NOT_PENDING (default) 1h: PLC_PENDING 1h: PLC_CLEAR
21	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PRC: 0h: PRC_NOT_PENDING (default) 1h: PRC_PENDING 1h: PRC_CLEAR
20	0	RW1C	T_XUSB_XHCI_OP_PORTSC_OCC: 0h: OCC_NOT_PENDING (default) 1h: OCC_PENDING 1h: OCC_CLEAR
19	0	RW1C	T_XUSB_XHCI_OP_PORTSC_WRC: 0h: WRC_NOT_PENDING (default) 1h: WRC_PENDING 1h: WRC_CLEAR
18	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PEC: 0h: PEC_NOT_PENDING (default) 1h: PEC_PENDING 1h: PEC_CLEAR
17	0	RW1C	T_XUSB_XHCI_OP_PORTSC_CSC: 0h: CSC_NOT_PENDING (default) 1h: CSC_PENDING 1h: CSC_CLEAR
16	0	RW1C	T_XUSB_XHCI_OP_PORTSC_LWS: 0h: LWS_DISABLED (default) 1h: LWS_ENABLED

Bits	Reset	R/W	Description
15:14	0	R/W	T_XUSB_XHCI_OP_PORTSC_PIC: 0h: PIC_OFF (default) 1h: PIC_AMBER 2h: PIC_GREEN 3h: PIC_UNDEFINED
13:10	0	R	T_XUSB_XHCI_OP_PORTSC_PSPD: 0h: PSPD_UNDEFINED (default) 1h: PSPD_FS 2h: PSPD_LS 3h: PSPD_HS 4h: PSPD_SS Fh: PSPD_UNKNOWN
9	1	R/W	T_XUSB_XHCI_OP_PORTSC_PP: 0h: PP_OFF 1h: PP_ON (default)
8:5	4h	RW1C	T_XUSB_XHCI_OP_PORTSC_PLS: 0h: PLS_U0 1h: PLS_U1 2h: PLS_U2 3h: PLS_U3 4h: PLS_DISABLED (default) 5h: PLS_RXDETECT 6h: PLS_INACTIVE 7h: PLS_POLLING 8h: PLS_RECOVERY 9h: PLS_HOTRESET Ah: PLS_COMPLIANCE Bh: PLS_LOOPBACK Fh: PLS_RESUME 0h: PLS_GOTO_U0 1h: PLS_GOTO_U1 2h: PLS_GOTO_U2 3h: PLS_GOTO_U3
4	0	R/W	T_XUSB_XHCI_OP_PORTSC_PR: 0h: PR_NOT_PENDING (default) 1h: PR_PENDING 1h: PR_SET
3	0	R	T_XUSB_XHCI_OP_PORTSC_OCA: 0h: OCA_FALSE (default) 1h: OCA_TRUE
2	0	R	T_XUSB_XHCI_OP_PORTSC_RSVD0: 0h: RSVD0_0 (default)
1	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PED: 0h: PED_DISABLED (default) 1h: PED_ENABLED 1h: PED_CLEAR
0	0	R	T_XUSB_XHCI_OP_PORTSC_CCS: 0h: CCS_NODEV (default) 1h: CCS_DEV

22.16.11.19 T_XUSB_XHCI_OP_PORTPMSCSS

Offset: 0x424 – 0x514 | Read/Write: R/W

Bits	Reset	R/W	Description
31:17	0	R	T_XUSB_XHCI_OP_PORTPMSCSS_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_FLA: 0h: FLA_INIT (default)
15:8	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_U2TIMEOUT: 0h: U2TIMEOUT_INIT (default)
7:0	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_U1TIMEOUT: 0h: U1TIMEOUT_INIT (default)

22.16.11.20 T_XUSB_XHCI_OP_PORTPMSCHS

Offset: 0x424 – 0x514 | Read/Write: R/W

Bits	Reset	R/W	Description
31:28	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_TM: 0h: TM_DISABLE (default) 1h: TM_JSTATE 2h: TM_KSTATE 3h: TM_SE0NAK 4h: TM_PACKET 5h: TM_FORCEEN Fh: TM_ERROR
27:17	0	R	T_XUSB_XHCI_OP_PORTPMSCHS_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_HLE: 0h: HLE_INIT (default)
15:8	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_L1DS: 0h: L1DS_INIT (default)
7:4	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_HIRD: 0h: HIRD_INIT (default)
3	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_RWE: 0h: RWE_DISABLED (default) 1h: RWE_ENABLED
2:0	0	R	T_XUSB_XHCI_OP_PORTPMSCHS_L1S: 0h: L1S_INVLD (default) 1h: L1S_SUCCESS 2h: L1S_NYET 3h: L1S_STALL 4h: L1S_ERROR

22.16.11.21 T_XUSB_XHCI_OP_PORTLISC

Offset: 0x428 – 0x458 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_OP_PORTLISC_RSVD0: 0h: RSVD0_00 (default)
15:0	0	R	T_XUSB_XHCI_OP_PORTLISC_LEC: 0h: LEC_INIT (default)

22.16.11.22 T_XUSB_XHCI_OP_PORTHLPMC

Offset: 0x42c – 0x51c | Read/Write: R

Bits	Reset	R/W	Description
13:10	0	R	T_XUSB_XHCI_OP_PORTHLPMC_BESLD: 0h: BESLD_INIT (default)
9:2	0	R	T_XUSB_XHCI_OP_ PORTHLPMC_L1_TIMEOUT: 0h: L1_TIMEOUT_INIT (default)
1:0	0	R	T_XUSB_XHCI_OP_ PORTHLPMC_HIRDM: 0h: HIRDM_INIT (default)

22.16.11.23 T_XUSB_XHCI_EC_USBLEGSUP

Offset: 0x600 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	T_XUSB_XHCI_EC_USBLEGSUP_RSVD1: 0h: RSVD1_00 (default)
24	0	R/W	T_XUSB_XHCI_EC_USBLEGSUP_OSSEM: 0h: OSSEM_INIT (default)

Bits	Reset	R/W	Description
23:17	0	R	T_XUSB_XHCI_EC_USBLEGSUP_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_EC_USBLEGSUP_BIOSSEM: 0h: BIOSSEM_INIT (default)
15:8	4h	R	T_XUSB_XHCI_EC_USBLEGSUP_NEXT: 4h: NEXT_SUPPROT_USB3 (default)
7:0	1h	R	T_XUSB_XHCI_EC_USBLEGSUP_CAPID: 1h: CAPID_USBLEGSUP (default)

22.16.11.24 T_XUSB_XHCI_EC_USBLEGCTLSTS

Offset: 0x604 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_BAR: 0h: BAR_NOT_PENDING (default) 1h: BAR_PENDING 1h: BAR_CLEAR
30	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_PCIC: 0h: PCIC_NOT_PENDING (default) 1h: PCIC_PENDING 1h: PCIC_CLEAR
29	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_OSOC: 0h: OSOC_NOT_PENDING (default) 1h: OSOC_PENDING 1h: OSOC_CLEAR
28:21	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD3: 0h: RSVD3_00 (default)
20	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_HSE: 0h: HSE_NOT_PENDING (default) 1h: HSE_PENDING
19:17	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD2: 0h: RSVD2_00 (default)
16	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_EVI: 0h: EVI_NOT_PENDING (default) 1h: EVI_PENDING
15	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_BAREN: 0h: BAREN_DISABLED (default) 1h: BAREN_ENABLED
14	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_PCIEN: 0h: PCIEN_DISABLED (default) 1h: PCIEN_ENABLED
13	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_OSOEN: 0h: OSOEN_DISABLED (default) 1h: OSOEN_ENABLED
12:5	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD1: 0h: RSVD1_00 (default)
4	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_HSEEN: 0h: HSEEN_DISABLED (default) 1h: HSEEN_ENABLED
3:1	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD0: 0h: RSVD0_00 (default)
0	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_SMIEN: 0h: SMIEN_DISABLED (default) 1h: SMIEN_ENABLED

22.16.11.25 T_XUSB_XHCI_EC_SUPPROT_USB3_0

Offset: 0x610 | Read/Write: R

Bits	Reset	R/W	Description
31:24	3h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_MAJORREV: 3h: MAJORREV_3 (default)
23:16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_MINORREV: 0h: MINORREV_0 (default)
15:8	4h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_NEXT: 4h: NEXT_SUPPROT_USB2 (default)
7:0	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_CAPID: 2h: CAPID_SUPPROT_USB3 (default)

22.16.11.26 T_XUSB_XHCI_EC_SUPPROT_USB3_1

Offset: 0x614 | Read/Write: R

Bits	Reset	R/W	Description
31:0	20425355h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_1_NAMESTR: 20425355h: NAMESTR_USB (default)

22.16.11.27 T_XUSB_XHCI_EC_SUPPROT_USB3_2

Offset: 0x618 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_RSVD0: 0h: RSVD0_00 (default)
15:8	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_PORTCNT:
7:0	1h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_PORTOFS: 1h: PORTOFS_VAL (default)

22.16.11.28 T_XUSB_XHCI_EC_SUPPROT_USB3_3

Offset: 0x61c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_3_SLOTTYPE: 0h: SLOTTYPE_VAL (default)

22.16.11.29 T_XUSB_XHCI_EC_SUPPROT_USB2_0

Offset: 0x620 | Read/Write: R

Bits	Reset	R/W	Description
31:24	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_MAJORREV: 2h: MAJORREV_3 (default)
23:16	0h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_MINORREV: 0h: MINORREV_0 (default)
15:8	4h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_NEXT: 4h: NEXT_DBCAP (default)
7:0	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_CAPID: 2h: CAPID_SUPPROT_USB3 (default)

22.16.11.30 T_XUSB_XHCI_EC_SUPPROT_USB2_1

Offset: 0x624 | Read/Write: R

Bits	Reset	R/W	Description
31:0	20425355h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_1_NAMESTR: 20425355h: NAMESTR_USB (default)

22.16.11.31 T_XUSB_XHCI_EC_SUPPROT_USB2_2

Offset: 0x628 | Read/Write: R

Bits	Reset	R/W	Description
31:28	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD2: 0h: RSVD2_00 (default)
27:25	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_MHD: 0h: MHD (default)
24:21	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD1: 0h: RSVD1_00 (default)
20	1	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_BLC: 1h: BLC_TRUE (default)
19	1	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_HLC: 1h: HLC_TRUE (default)
18	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_IHI: 0h: IHI_TRUE (default)
17	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_HSO: 0h: HSO_TRUE (default)
16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD0: 0h: RSVD0_00 (default)
15:8	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_PORTCNT:
7:0	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_PORTOFS:

22.16.11.32 T_XUSB_XHCI_EC_SUPPROT_USB2_3

Offset: 0x62c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_3_SLOTTYPE: 0h: SLOTTYPE_VAL (default)

22.16.11.33 T_XUSB_XHCI_EC_DBCAP_DCID

Offset: 0x630 | Read/Write: R

Bits	Reset	R/W	Description
31:21	0	R	Reserved
20:16	1h	R	T_XUSB_XHCI_EC_DBCAP_DCID_DCERSTM: 1h: DCERSTM_VALUE (default)
15:8	0	R	T_XUSB_XHCI_EC_DBCAP_DCID_NEXT: 0h: NEXT_NONE (default)
7:0	Ah	R	T_XUSB_XHCI_EC_DBCAP_DCID_CAPID: Ah: CAPID_DBCAP (default)

22.16.11.34 T_XUSB_XHCI_EC_DBCAP_DCDB

Offset: 0x634 | Read/Write: W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_DBCAP_DCDB_RSVD1: 0h: RSVD1_00 (default)
15:8	0	W	T_XUSB_XHCI_EC_DBCAP_DCDB_DBTARGET: 0h: DBTARGET_INIT (default)
7:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCDB_RSVD0: 0h: RSVD0_00 (default)

22.16.11.35 T_XUSB_XHCI_EC_DBCAP_DCERSTSZ

Offset: 0x638 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_DBCAP_DCERSTSZ_RSVD0: 0h: RSVD0_00 (default)
15:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTSZ_ERSTSZ: 0h: ERSTSZ_INIT (default)

22.16.11.36 T_XUSB_XHCI_EC_DBCAP_RSVD0

Offset: 0x63c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_DBCAP_RSVD0_RSVD0: 0h: RSVD0_00 (default)

22.16.11.37 T_XUSB_XHCI_EC_DBCAP_DCERSTBALO

Offset: 0x640 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTBALO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCERSTBALO_RSVD0: 0h: RSVD0_00 (default)

22.16.11.38 T_XUSB_XHCI_EC_DBCAP_DCERSTBAHI

Offset: 0x644 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTBAHI_ADDRHI: 0h: ADDRHI_INIT (default)

22.16.11.39 T_XUSB_XHCI_EC_DBCAP_DCERDPLO

Offset: 0x648 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3	0	R	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_RSVD0: 0h: RSVD0_00 (default)
2:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_DESI: 0h: DESI_INIT (default)

22.16.11.40 T_XUSB_XHCI_EC_DBCAP_DCERDPHI

Offset: 0x64c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPHI_ADDRHI: 0h: ADDRHI_INIT (default)

22.16.11.41 T_XUSB_XHCI_EC_DBCAP_DCCTRL

Offset: 0x650 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DCE: 0h DCE_DIS (default) 1h DCE_EN
30:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DEVADR: 0h: DEVADR_INIT (default)
23:16	15h	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_MAXBURST: 15h: MAXBURST_INIT (default)
15:5	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_RSVD0: 0h: RSVD0_00 (default)
4	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DRC: 0h: DRC_INIT (default) 1h: DRC_SET 1h: DRC_CLEAR
3	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_HIT: 0h: HIT_FALSE (default) 1h: HIT_TRUE
2	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_HOT: 0h: HOT_FALSE (default) 1h: HOT_TRUE
1	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_LSE: 0h: LSE_DIS (default) 1h: LSE_EN
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DCR: 0h: DCR_STOP (default) 1h: DCR_RUN

22.16.11.42 T_XUSB_XHCI_EC_DBCAP_DCST

Offset: 0x654 | Read/Write: R

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_DPN: 0h: DPN_INIT (default)
23:1	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_RSVD: 0h: RSVD_00 (default)
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_ER: 0h: ER_EMPTY (default) 1h: ER_NOTEMPTY

22.16.11.43 T_XUSB_XHCI_EC_DBCAP_DCPORTSC

Offset: 0x658 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD4: 0h: RSVD4_00 (default)

Bits	Reset	R/W	Description
23	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CEC: 0h: CEC_NOT_PENDING (default) 1h: CEC_PENDING 1h: CEC_CLEAR
22	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PLC: 0h: PLC_NOT_PENDING (default) 1h: PLC_PENDING 1h: PLC_CLEAR
21	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PRC: 0h: PRC_NOT_PENDING (default) 1h: PRC_PENDING 1h: PRC_CLEAR
20:18	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD3: 0h: RSVD3_00 (default)
17	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CSC: 0h: CSC_NOT_PENDING (default) 1h: CSC_PENDING 1h: CSC_CLEAR
16:14	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD2: 0h: RSVD2_00 (default)
13:10	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PS: 0h: PS_UNDEFINED (default) 4h: PS_SS
9	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD1: 0h: RSVD1_00 (default)
8:5	4h	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PLS: 0h: PLS_U0 1h: PLS_U1 2h: PLS_U2 3h: PLS_U3 4h: PLS_DISABLED (default) 5h: PLS_RXDETECT 6h: PLS_INACTIVE 7h: PLS_POLLING 8h: PLS_RECOVERY 9h: PLS_HOTRESET Ah: PLS_COMPLIANCE Bh: PLS_LOOPBACK Fh: PLS_RESUME
4	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PR: 0h: PR_NORST (default) 1h: PR_RST
3:2	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD0: 0h: RSVD0_00 (default)
1	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PED: 0h: PED_DIS (default) 1h: PED_EN
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CCS: 0h: CCS_NOCON (default) 1h: CCS_CON

22.16.11.44 T_XUSB_XHCI_EC_DBCAP_RSVD1

Offset: 0x65c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_DBCAP_RSVD1_RSVD0: 0h: RSVD0_00 (default)

22.16.11.45 T_XUSB_XHCI_EC_DBCAP_DCECPLO

Offset: 0x660 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCECPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCECPLO_RSVD0: 0h: RSVD0_00 (default)

22.16.11.46 T_XUSB_XHCI_EC_DBCAP_DCECPHI

Offset: 0x664 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCECPHI_ADDRHI: 0h: ADDRHI_INIT (default)

22.16.11.47 T_XUSB_XHCI_EC_DBCAP_INFO0

Offset: 0x668 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO0_VENDORID: 0h: VENDORID_INIT (default)
15:8	0	R	Reserved
7:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO0_PROTOCOL: 0h: PROTOCOL_VENDOR (default) 1h: PROTOCOL_GNU

22.16.11.48 T_XUSB_XHCI_EC_DBCAP_INFO1

Offset: 0x66c | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO1_DEV_REV: 0h: DEV_REV_INIT (default)
15:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO1_PRODUCTID: 0h: PRODUCTID_INIT (default)

22.16.11.49 T_XUSB_XHCI_RT_MFINDEX

Offset: 0x800 | Read/Write: R

Bits	Reset	R/W	Description
31:14	0	R	T_XUSB_XHCI_RT_MFINDEX_RSVD0: 0h: RSVD0_00
13:0	Unknown	R	T_XUSB_XHCI_RT_MFINDEX_MFINDEX:

22.16.11.50 T_XUSB_XHCI_RT_IMAN

Offset: 0x820 | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R	T_XUSB_XHCI_RT_IMAN_RSVD0: 0h: RSVD0_00
1	0	R/W	T_XUSB_XHCI_RT_IMAN_IE: 0h: IE_DISABLED (default) 1h: IE_ENABLED

Bits	Reset	R/W	Description
0	0	RW1C	T_XUSB_XHCI_RT_IMAN_IP: 0h: IP_NOT_PENDING (default) 1h: IP_PENDING 1h: IP_CLEAR

22.16.11.51 T_XUSB_XHCI_RT_IMOD

Offset: 0x824 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	Unknown	R/W	T_XUSB_XHCI_RT_IMOD_IMODC:
15:0	0FA0h	R/W	T_XUSB_XHCI_RT_IMOD_IMODI: FA0h: IMODI_INIT (default)

22.16.11.52 T_XUSB_XHCI_RT_ERSTSZ

Offset: 0x828 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_RT_ERSTSZ_RSVD0: 0h: RSVD0_00
15:0	0	R/W	T_XUSB_XHCI_RT_ERSTSZ_SZ: 0h: SZ_INIT (default)

22.16.11.53 T_XUSB_XHCI_RT_ERRSVD

Offset: 0x82c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_RT_ERRSVD_RSVD0: 0h: RSVD0_00

22.16.11.54 T_XUSB_XHCI_RT_ERSTBA0

Offset: 0x830 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_RT_ERSTBA0_ERSTBLO: 0h: ERSTBLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_RT_ERSTBA0_RSVD0: 0h: RSVD0_00

22.16.11.55 T_XUSB_XHCI_RT_ERSTBA1

Offset: 0x834 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_RT_ERSTBA1_ERSTBHI: 0h: ERSTBHI_INIT (default)

22.16.11.56 T_XUSB_XHCI_RT_ERDP0

Offset: 0x838 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_RT_ERDP0_DQPTRLO: 0h: DQPTRLO_INIT (default)
2:0	0	R/W	T_XUSB_XHCI_RT_ERDP0_DESI: 0h: DESI_INIT (default)

22.16.11.57 T_XUSB_XHCI_RT_ERDP1

Offset: 0x83c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_RT_ERDP1_DQPTRHI: 0h: DQPTRHI_INIT (default)

22.16.11.58 T_XUSB_XHCI_DB

Offset: 0xc00 – 0xffc | Read/Write: W

Bits	Reset	R/W	Description
31:16	0	W	T_XUSB_XHCI_DB_STREAMID: 0h: STREAMID_INIT (default)
15:8	0	R	T_XUSB_XHCI_DB_RSVD0: 0h: RSVD0_00
7:0	0	W	T_XUSB_XHCI_DB_TARGET: 0h: TARGET_INIT (default)

22.16.12 XUSB CSB Registers

22.16.12.1 XUSB_CSB_MEMPOOL_ILOAD_ATTR_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a00 | Read/Write: R/W | Reset: 0b000xxxxxxxxx00000000000000000000

Bit	R/W	Reset	Description
31	RW	0x0	TC 0: TC_DEFAULT
30	RW	0x0	NS 0: NS_DEFAULT
29	RW	0x0	RO 0: RO_DEFAULT
28	RW	0x0	RDPASSPW 0: RDPASSPW_DEFAULT
19:8	RW	0x0	SIZE 0: SIZE_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

22.16.12.2 XUSB_CSB_MEMPOOL_ILOAD_BASE_LO_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a04 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	R/W	Reset	Description
31:8	RW	0x0	RSVD 0: RSVD_DEFAULT
7:0	RO	0x0	SRC_ADDR 0: SRC_ADDR_DEFAULT

22.16.12.3 XUSB_CSB_MEMPOOL_ILOAD_BASE_HI_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a08 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
7:0	0x0	SRC_ADDR 0: SRC_ADDR_DEFAULT

22.16.12.4 XUSB_CSB_MEMPOOL_L2IMEMOP_SIZE_0

L2IMEMOP Operational Register

Offset: 0x0101a10 | Read/Write: R/W | Reset: 0b0000000xxxxx00000000000000000000

Bit	R/W	Reset	Description
31:24	RW	0x0	SRC_COUNT 0: SRC_COUNT_DEFAULT
19:8	RW	0x0	SRC_OFFSET 0: SRC_OFFSET_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

22.16.12.5 XUSB_CSB_MEMPOOL_L2IMEMOP_TRIG_0

L2IMEMOP Operational Register

L2IMEMop Action encodings

bit[0]: 1=> RESULT, 0=> NO RESULT

bit[1]: 1=> OPTIMIZED, 0=> UNOPTIMIZED

bit[7:4] 0001 => LOAD_LOCKED

0010 => UNLOCK

0100 => INVALIDATE_ALL

0101 => QUERY_INDEX

0110 => QUERY_SPACE

Offset: 0x0101a14 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx000000000000000000

Bit	R/W	Reset	Description
31:24	RW	X	ACTION 0x10: L2IMEM_LOAD_LOCKED 0x11: L2IMEM_LOAD_LOCKED_RESULT 0x20: L2IMEM_UNLOCK 0x21: L2IMEM_UNLOCK_RESULT 0x22: L2IMEM_UNLOCK_OPTIMIZED 0x23: L2IMEM_UNLOCK_OPTIMIZED_RESULT 0x40: L2IMEM_INVALIDATE_ALL 0x41: L2IMEM_INVALIDATE_ALL_RESULT 0x50: L2IMEM_QUERY_INDEX_RESULT 0x60: L2IMEM_QUERY_SPACE_RESULT 0xFF: WAITFENCE_RESULT
17:8	RW	0x0	DEST_INDEX 0: DEST_INDEX_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

22.16.12.6 XUSB_CSB_MEMPOOL_APMAP_0

Aperture Programming Register

Offset: 0x010181c | Read/Write: R/W | Reset: 0b1xxxxxxxxxxx0xxxxxxxxxxxxxx00xxxxx000

Bit	Reset	Description
31	0x1	BOOTPATH 1: BOOTPATH_DEFAULT
24	0x0	XREQ_READ 0: XREQ_READ_DEFAULT
9:8	0x0	XMAP 0: XMAP_A 1: XMAP_B 2: XMAP_C
2:0	0x0	FDDMA 0: FDDMA_A 1: FDDMA_B 2: FDDMA_C 3: FDDMA_D 4: FDDMA_E

22.16.12.7 FALCON_CPUCTL_0

Offset: 0x100 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	R/W	Description
5	X	RO	STOPPED: This bit indicates whether the CPU is currently in the stopped state. The Falcon processor will exit this state if a 1 is written to the STARTCPU bit, or if an interrupt arrives on one of its two inputs and the corresponding IE bit in CSW is set 0: STOPPED_FALSE 1: STOPPED_TRUE
4	X	RO	HALTED: This bit indicates whether the CPU is currently in the halted state. The Falcon processor can only exit this state when a 1 is written to the STARTCPU bit. 0: HALTED_FALSE 1: HALTED_TRUE
3	X	WO	HRESET: Setting HRESET to true will apply a hard reset. This bit will auto-clear and setting to false has no effect. 0: HRESET_FALSE 1: HRESET_TRUE
2	X	WO	SRESET: Setting SRESET to true will apply a soft reset. This bit will auto-clear and setting to false has no effect. 0: SRESET_FALSE 1: SRESET_TRUE
1	X	WO	STARTCPU: Setting STARTCPU to true will start CPU execution while in a HALTED state. If a start request is still pending, setting to false cancel the start request. Writing any value has no effect while the CPU is running. 0: STARTCPU_FALSE 1: STARTCPU_TRUE
0	X	WO	IINVAL: Setting IINVAL to true causes all blocks in IMEM except block 0 to be marked as INVALID. This bit will auto-clear and setting to false has no effect. 0: IINVAL_FALSE 1: IINVAL_TRUE

22.16.12.8 FALCON_BOOTVEC_0

The BOOTVEC register stores the initial execution start address of the CPU when it is first started after a reset.

Offset: 0x104 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VEC 0: VEC_INIT

22.16.12.9 FALCON_DMACTL_0

Offset: 0x10c | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxxx1

Bit	Reset	R/W	Description
6:3	X	RO	DMAQ_NUM: Indicates the valid request number at the DMA request queue
2	X	RO	IMEM_SCRUBBING: 0: Indicates scrubbing is done. For a non-secure Falcon, this value is always 0. 1: Indicates secure scrubber is pending and scrubbing IMEM, any access to IMEM will be blocked until scrubbing is done 0: IMEM_SCRUBBING_DONE 1: IMEM_SCRUBBING_PENDING
1	X	RO	DMEM_SCRUBBING: 0: Indicates scrubbing is done. For a non-secure Falcon, this value is always 0. 1: Indicates secure scrubber is pending and scrubbing DMEM, any access to DMEM will be blocked until scrubbing is done 0: DMEM_SCRUBBING_DONE 1: DMEM_SCRUBBING_PENDING
0	0x1	RW	REQUIRE_CTX: When REQUIRE_CTX is set to true, a valid context must be loaded before any DMA request can be serviced. Pending requests without a valid current context remain pending, and do not prevent the engine from reporting idle. When this bit is set to false, DMA requests are serviced regardless of the current context. Note that once a request is issued, it must complete before the engine will be able to report idle, as needed for example to process WFI context switch requests. 0:REQUIRE_CTX_FALSE 1: REQUIRE_CTX_TRUE 1: REQUIRE_CTX_INIT

22.16.12.10 FALCON_IMFILLRNG1_0

IMFILLRNG1 indicates tag values for the low and high end of the PC range to be auto-filled. The PC range is [tag_lo<<8. tag_hi<<8+255].

If the user enables the auto-fill feature and leaves IMFILLRNG1 as zero, instruction 0x0~0x0ff always works as auto-fill range.

Offset: 0x154 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TAG_HI 0: TAG_HI_INIT
15:0	0x0	TAG_LO TAG_LO_INIT

22.16.12.11 FALCON_IMFILLCTL_0

Offset: 0x158 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
7:0	0x0	NBLOCKS 0: NBLOCKS_INIT

22.16.13 XUSB Device Registers

Base Address: XUSB_DEV_BASE + 0x9000

22.16.13.1 XUSB_DEV_AXI_BAR0_SZ_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxx00000000000000001000)

Bit	Reset	Description
19:0	0x8	AXI_BAR0_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.13.2 XUSB_DEV_AXI_BAR1_SZ_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR1_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.13.3 XUSB_DEV_AXI_BAR2_SZ_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR2_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.13.4 XUSB_DEV_AXI_BAR3_SZ_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR3_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

22.16.13.5 XUSB_DEV_AXI_BAR0_START_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x700d0000 (0b01110000000011010000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x700d0	AXI_BAR0_START:The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.13.6 XUSB_DEV_AXI_BAR1_START_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR1_START:The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.13.7 XUSB_DEV_AXI_BAR2_START_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR2_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.13.8 XUSB_DEV_AXI_BAR3_START_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR3_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

22.16.13.9 XUSB_DEV_FPCI_BAR0_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00700d01 (0b00000000011100000000011010000xxx1)

Bit	Reset	Description
31:4	0x700d0	FPCI_BAR0_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this register.
0	0x1	FPCI_BAR0_ACCESS_TYPE:Indicates if the address region is memory mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=IO/config access (NPW only)

22.16.13.10 XUSB_DEV_FPCI_BAR1_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR1_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this register.
0	0x1	FPCI_BAR1_ACCESS_TYPE: Indicates if the address region is memory-mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=IO/config access (NPW only)

22.16.13.11 XUSB_DEV_FPCI_BAR2_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR2_START:The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this register.
0	0x1	FPCI_BAR2_ACCESS_TYPE:Indicates if the address region is memory-mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=IO/config access (NPW only)

22.16.13.12 XUSB_DEV_FPCI_BAR3_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR3_START:The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this register.
0	0x1	FPCI_BAR3_ACCESS_TYPE:Indicates if the address region is memory-mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=IO/config access (NPW only)

22.16.13.13 XUSB_DEV_MSI_BAR_SZ_0

MSI BAR SIZE

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE: The size of the address range associated with MSI BAR is in 4K increments. Value of 0 signifies BAR is not used.

22.16.13.14 XUSB_DEV_MSI_AXI_BAR_ST_0

MSI AXI BAR START

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	MSI_AXI_BAR_START: The start of upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:12of MSI BAR start correspond to AXI address bits 31:12.

22.16.13.15 XUSB_DEV_MSI_FPCI_BAR_ST_0

MSI FPCI BAR START

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xxxx)

Bit	Reset	Description
31:4	0x0	MSI_FPCI_BAR_START: The start of upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4of MSI BAR start correspond to UFPCI address bits 39:12.

22.16.13.16 XUSB_DEV_MSI_VEC0_0

MSI VECTORi, i in [0,7], RW

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR0:Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.17 XUSB_DEV_MSI_VEC1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR1:Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.18 XUSB_DEV_MSI_VEC2_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR2:Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.19 XUSB_DEV_MSI_VEC3_0

Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR3:Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.20 XUSB_DEV_MSI_VEC4_0

Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR4:Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.21 XUSB_DEV_MSI_VEC5_0

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR5: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.22 XUSB_DEV_MSI_VEC6_0

Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR6: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.23 XUSB_DEV_MSI_VEC7_0

Offset: 0x11c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR7:Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

22.16.13.24 XUSB_DEV_MSI_EN_VEC0_0

MSI ENABLE VECTORi, i in [0,7], RW

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR0: Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.25 XUSB_DEV_MSI_EN_VEC1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR1: Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.26 XUSB_DEV_MSI_EN_VEC2_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR2:Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.27 XUSB_DEV_MSI_EN_VEC3_0

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR3: Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.28 XUSB_DEV_MSI_EN_VEC4_0

Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR4: Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.29 XUSB_DEV_MSI_EN_VEC5_0

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR5:Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.30 XUSB_DEV_MSI_EN_VEC6_0

Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR6: Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.31 XUSB_DEV_MSI_EN_VEC7_0

Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR7:Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

22.16.13.32 XUSB_DEV_CONFIGURATION_0

Configuration

Offset: 0x180 | Read/Write: R/W | Reset: 0x800X8X40 (0b1xxxxxxxxxxx1xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x1	CLKEN_OVERRIDE:This can override the clock enable in case of malfunction.
19	RW	0x1	PW_NO_DEVSEL_ERR_CYA: Set this to not detect DECERR due to no DEVSEL for DS PWs only.
18	RO	X	INITIATOR_READ_IDLE:This read-only bit provides status reads on AFI upstream. A value of 1b indicates there are no outstanding reads to initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on AFI upstream. A value of 1b indicates there are no outstanding writes to initiator.
15	RW	0x1	WDATA_LEAD_CYA: Used to enable/disable the handling of write data ahead of requests on IPFS axi target
14	RW	0x0	WR_INTRLV_CYA: Used to enable/disable the handling of interleaved write requests on IPFS AXI target
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to IPFS target. A value of 1b indicates there are no outstanding reads to downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to IPFS target. A value of 1b indicates there are no outstanding writes to downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any active bits valid or not
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = whenever MSI is ready assert the interrupt 0 = default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: Input to upstream FPCI 1 = whenever write is ready, send it; 0 = write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to upstream FPCI. Allow upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for upstream FPCI. Allow upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: Used for downstream FPCI. Allow downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: Input to downstream FPCI. Allow downstream FPCI responses to pass writes
1	RW	0x0	DFPCI_PASSPW: Input to downstream FPCI. Allow downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the IPFS device block is disabled, it is completely invisible on the IPFS bus, i.e. it doesn’t even process IPFS configuration accesses.

22.16.13.33 XUSB_DEV_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Master Abort. 1 = forward error, 0 = return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Data Error. 1 = forward error, 0 = return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Target Abort. This bit also covers decode error generated when there is no devsel received. 1 = forward error, 0 = return AXI OKAY response (2'b0)

22.16.13.34 XUSB_DEV_INTR_MASK_0

Interrupt Masks

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxx0xxxxxxx0)

Bit	Reset	Description
16	0x0	IP_INT_MASK:IP (SATA/AZA) interrupt to MPCORE gated by mask.
8	0x0	MSI_MASK:MSI to MPCORE gated by mask.
0	0x0	INT_MASK: Interrupt to MPCORE gated by mask.

22.16.13.35 XUSB_DEV_INTR_CODE_0

Interrupt Control

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: Eight interrupt codes If the code is 0, logging of the next interrupt is enabled 0 = INT_CODE_CLEAR: Clear interrupt code 1 = INT_CODE_INI_SLVERR: Interrupt code for MPCORE AXI SLVERR response to IPFS 2 = INT_CODE_INI_DECERR: Interrupt code for MPCORE AXI DECERR response to IPFS 3 = INT_CODE_TGT_SLVERR: Interrupt code for PCIe endpoint FPCI target abort or data error response to IPFS 4 = INT_CODE_TGT_DECERR: Interrupt code for PCIe2 FPCI master abort response to IPFS 5 = INT_CODE_TGT_WRERR: Interrupt code for bufferable write to non-posted write address region 6 = RSVD1 : Reserved 7 = INT_CODE_DFPCI_DECERR: Interrupt code for PCIe2 response to downstream request when downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR: Interrupt code for IPFS response to downstream request when AXI target AXI address does not fall in any of IPFS downstream BARs 9 = INT_CODE_FPCI_TIMEOUT: Interrupt code for FPCI Timeout 10 = RSVD2: Reserved for future expansion 11 = RSVD3 12 = RSVD4 13 = RSVD5 14 = RSVD6 15 = INT_CODE_SM_FATAL_ERROR: Interrupt code for SM fatal error 16 = INT_CODE_SM_NON_FATAL_ERROR: Interrupt code for SM non-fatal error

22.16.13.36 XUSB_DEV_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, it contains address bits [31:2], either in FPCI memory space or AXI space. For FPCI generated errors, the information contains FPCI address. For AXI/IPFS generated errors, the information contains AXI address.
0	0x0	DIR: Indicates direction of the AXI/FPCI transaction. 1=rd/0=wr. If signature type is 6 (sideband message), this field is 1. 0 = WRITE: Interrupt due to a write transaction 1 = READ: Interrupt due to a read transaction

22.16.13.37 XUSB_DEV_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of captured FPCI address (bits[39:32])when interrupt code is 3, 4, or 7. These bits determine the region in the Hypertransport Address Map that was accessed.

22.16.13.38 XUSB_DEV_IPFS_INTR_ENABLE_0

IPFS Interrupt Enable

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxx00000000)

Bit	Reset	Description
13	0x0	EN_SM_NON_FATAL_ERROR: Enable bit for interrupt code 15
12	0x0	EN_SM_FATAL_ERROR: Enable bit for interrupt code 14
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

22.16.13.39 XUSB_DEV_UFPCI_CONFIG_0

Upstream FPCI Configuration

Offset: 0x19c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00010)

Bit	Reset	Description
4:0	0x2	UNITID_T0C0:Upstream FPCI Unit ID for controller 0. HyperTransport, upstream FPCI request

22.16.13.40 XUSB_DEV_CFG_REVID_0

CFG_REVID register

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x000X10XX (0bxxxxxxxxxxxxxxxxxx0100xxxxxxx1xx)

Bit	R/W	Reset	Description
19	RO	X	DEV2SM_NONISO_REQUEST_PEND: This is to tell if there is a non ISO request pending. 0 = NO 1 = YES
18	RO	X	DEV2SM_ISO_REQUEST_PEND: This is to tell if there is an ISO request pending. 0 = NO 1 = YES
13:12	RW	0x1	STRAP_CPU_MODE: MCP: mode to send MSI. Can have it programmable 0 = NB_INTEL 1 = NB_AMD 2 = AMD 3 = TMTA
11	RW	0x0	CFG_REVID_WRITE_ENABLE: MCP: the enable to override the REVID. Can have it programmable 0 = CLEAR 1 = SET
10	RW	0x0	CFG_REVID_OVERRIDE: MCP: a way to override the current revision ID. Can have it programmable 0 = DISABLE 1 = ENABLE
4	RO	X	DEV2LEG_NONCOH_REQUEST_PEND: MCP: Tells the leg block that we have a non coherent request pending. 0 = NO 1 = YES
3	RO	X	DEV2LEG_COH_REQUEST_PEND: MCP comment: Tells the leg block that we have a coherent request pending 0 = NO 1 = YES

Bit	R/W	Reset	Description
2	RW	0x1	SM2DEV_FPCI_TIMEOUT_EN: FPCI timeout enable bit for Controller 0: disable 1: enable 0 = DISABLE 1 = ENABLE

22.16.13.41 XUSB_DEV_FPCI_TIMEOUT_0

FPCI_TIMEOUT register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x000f0000 (0bxxxxxxxxxxxx11110000000000000000)

Bit	Reset	Description
19:0	0xf0000	SM2ALL_FPCI_TIMEOUT_THRESH: This sets the timeout threshold value for the FPCI bus. The starts counting for each queues (ISO/NISO- RD/WR) have pending request in FPCI wrapper, the count resets when the requests popped.

22.16.13.42 XUSB_DEV_TOM_0

Top of Memory Limit

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x3fff0fff (0bxx11111111111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3fff	LEG2ALL_TOM2:Top of Memory Limit 2.
11:0	0xfff	LEG2ALL_TOM1:Top of Memory Limit 1.

22.16.13.43 XUSB_DEV_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x1ac | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND: Number of pending initiator ISO PW responses

22.16.13.44 XUSB_DEV_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x1b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND: Number of pending initiator NISO PW responses

22.16.13.45 XUSB_DEV_INTR_STATUS_0

IPFS Interrupt Status

Offset: 0x1b4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	IP_INTR_STATUS: status of IP (SATA/AZA) interrupt
1	X	MSI_INTR_STATUS: status of MSI interrupt
0	X	IPFS_INTR_STATUS: status of IPFS interrupt

22.16.13.46 XUSB_DEV_DFPCI_BEN_0

Downstream FPCI Byte Enables

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	0x0	EN_DFPCI_BEN: Enable bit for BEN; when set, programmed BE is sent on DFPCI bus
3:0	0x0	DFPCI_BYTE_ENABLE_N: Active low byte enables

22.16.13.47 XUSB_DEV_CLKGATE_HYSTERESIS_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000014 (0bxxxxxxxxxxxxxxxxxxxxxxxx00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of IPFS clock cycles to wait after clock gating criteria is met to disable IPFS/ FPCI clocks

22.16.13.48 XUSB_DEV_XUSB_DEV_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Note: *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility.*

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clk is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC.

A '1' written to the cclk override field keeps client's clk always on inside MCCIF.

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	XUSB_DEV_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	XUSB_DEV_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	XUSB_DEV_CCLK_OVERRIDE
17	0x0	XUSB_DEV_RCLK_OVERRIDE
16	0x0	XUSB_DEV_WCLK_OVERRIDE
3	DISABLE	XUSB_DEV_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	XUSB_DEV_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	XUSB_DEV_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	XUSB_DEV_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

22.16.13.49 XUSB_DEV_ORDERING_RULES_0

Offset: 0x1e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000x)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW: Modified Upstream MSIAW ordering. 0 = Tegra X1 MSIAW behavior 1 = Tegra 3 MSIAW behavior
2	0x0	UPSTREAM_RESPAW: Modified RespAW ordering. 0 = Tegra X1 RespAW behavior 1 = Tegra 3 RespAW behavior
1	0x0	UPSTREAM_RAW: Modified RAW ordering. 0 = Tegra X1 RAW behavior 1 = Tegra 3 RAW behavior

22.16.13.50 XUSB_DEV_A2F_UFPCI_CFG0_0

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

22.16.13.51 XUSB_DEV_A2F_UFPCI_CFG1_0

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

22.16.14 XUSB Device PCI Config Registers

Base Address: XUSB_DEV_BASE + 0x8000

22.16.14.1 _T_XUSB_DEV_CFG_0

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:16	FADh	R	T_XUSB_DEV_CFG_0_DEVICE_ID_UNIT: FA3h: DEVICE_ID_UNIT_XUSB (default)
15:0	10DEh	R	T_XUSB_DEV_CFG_0_VENDOR_ID: 10DEh: VENDOR_ID_NVIDIA (default)

22.16.14.2 T_XUSB_DEV_CFG_1

FPCI Configuration Register

Offset: 0x04 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	T_XUSB_DEV_CFG_1_DETECTED_PERR: 0h: DETECTED_PERR_NOT_ACTIVE (default) 1h: DETECTED_PERR_ACTIVE 1h: DETECTED_PERR_CLEAR
30	0	R	T_XUSB_DEV_CFG_1_SIGNALED_SERR: 0h: SIGNALED_SERR_NOT_ACTIVE (default) 1h: SIGNALED_SERR_ACTIVE 1h: SIGNALED_SERR_CLEAR
29	0	R	T_XUSB_DEV_CFG_1_RECEIVED_MASTER: 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_CLEAR
28	0	R	T_XUSB_DEV_CFG_1_RECEIVED_TARGET: 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_CLEAR
27	0	R	T_XUSB_DEV_CFG_1_SIGNALED_TARGET: 0h: SIGNALED_TARGET_NO_ABORT (default) 1h: SIGNALED_TARGET_ABORT 1h: SIGNALED_TARGET_CLEAR
26:25	0	R	XUSB_DEV_CFG_1_DEVSEL_TIMING 0h: XUSB_DEV_CFG_1_DEVSEL_TIMING_FAST 1h: XUSB_DEV_CFG_1_DEVSEL_TIMING_MEDIUM 2h: XUSB_DEV_CFG_1_DEVSEL_TIMING_SLOW
24	0	R	T_XUSB_DEV_CFG_1_MASTER_DATA_PERR: 0h: MASTER_DATA_PERR_NOT_ACTIVE (default) 1h: MASTER_DATA_PERR_ACTIVE 1h: MASTER_DATA_PERR_CLEAR
23	1h	R	T_XUSB_DEV_CFG_1_FAST_BACK2BACK: 0h: FAST_BACK2BACK_INCAPABLE 1h: FAST_BACK2BACK_CAPABLE (default)
22	0	R	Reserved
21	1h	R	T_XUSB_DEV_CFG_1_66 MHZ: 0h: 66 MHZ_INCAPABLE 1h: 66 MHZ_CAPABLE (default)
20	1h	R	T_XUSB_DEV_CFG_1_CAPLIST: 0h: CAPLIST_NOT_PRESENT 1h: CAPLIST_PRESENT (default)
19	None	R	T_XUSB_DEV_CFG_1_INTR_STATUS: 0h: INTR_STATUS_0 1h: INTR_STATUS_1
18:11	0	R	Reserved
10	0	R/W	T_XUSB_DEV_CFG_1_INTR_DISABLE: 0h: INTR_DISABLE_ON (default) 1h: INTR_DISABLE_OFF
9	0	R	T_XUSB_DEV_CFG_1_BACK2BACK: 0h: BACK2BACK_DISABLED (default) 1h: BACK2BACK_ENABLED
8	0	R/W	T_XUSB_DEV_CFG_1_SERR: 0h: SERR_DISABLED (default) 1h: SERR_ENABLED
7	0	R	T_XUSB_DEV_CFG_1_STEP: 0h: STEP_DISABLED (default) 1h: STEP_ENABLED
6	0	R/W	T_XUSB_DEV_CFG_1_PERR: 0h: PERR_DISABLED (default) 1h: PERR_ENABLED

Bits	Reset	R/W	Description
5	0	R	T_XUSB_DEV_CFG_1_PALETTE_SNOOP: 0h: PALETTE_SNOOP_DISABLED (default) 1h: PALETTE_SNOOP_ENABLED
4	0	R	T_XUSB_DEV_CFG_1_WRITE_AND_INVAL: 0h: WRITE_AND_INVAL_DISABLED (default) 1h: WRITE_AND_INVAL_ENABLED
3	0	R	T_XUSB_DEV_CFG_1_SPECIAL_CYCLE: 0h: SPECIAL_CYCLE_DISABLED (default) 1h: SPECIAL_CYCLE_ENABLED
2	0	R/W	T_XUSB_DEV_CFG_1_BUS_MASTER: 0h: BUS_MASTER_DISABLED (default) 1h: BUS_MASTER_ENABLED
1	0	R/W	T_XUSB_DEV_CFG_1_MEMORY_SPACE: 0h: MEMORY_SPACE_DISABLED (default) 1h: MEMORY_SPACE_ENABLED
0	0	R/W	T_XUSB_DEV_CFG_1_IO_SPACE: 0h: IO_SPACE_DISABLED (default) 1h: IO_SPACE_ENABLED

22.16.14.3 T_XUSB_DEV_CFG_2

PCI Revision ID and Class Code Register

Offset: 0x08 | Read/Write: R

Bits	Reset	R/W	Description
31:24	Ch	R	T_XUSB_DEV_CFG_2_BASE_CLASS: The CLASS_CODE bits identify the generic function of the device and (in some cases) a specific register-level programming interface. The register is broken into three byte-size fields. The upper byte (at offset 0BH) is a base class code which broadly classifies the type of function the device performs. The middle-byte (at offset 0BH) is a sub-class code which identifies more specifically the function of the device. The lower byte (at offset 09H) identifies a specific register-level programming interface (if any) so that device independent software can interact with the device. The Class Code and Revision ID are defined by parameters per block. Ch: BASE_CLASS_SBC (default)
23:16	3h	R	T_XUSB_DEV_CFG_2_SUB_CLASS: 3h: SUB_CLASS_XUSB (default)
15:8	FEh	R	T_XUSB_DEV_CFG_2_PROG_IF: 30h: PROG_IF_XHCI (default)
7:0	A1h	R	T_XUSB_DEV_CFG_2_REVISION_ID: The REVISION_ID bits specify a device specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. This field should be viewed as a vendor defined extension to the DEVICE_ID. A1h: REVISION_ID_VAL (default) A1h: REVISION_ID_A01

22.16.14.4 T_XUSB_DEV_CFG_3

PCI Configuration Register

Offset: 0x0C | Read/Write: R

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23	0	R	T_XUSB_DEV_CFG_3_HEADER_TYPE_FUNC: 0h: HEADER_TYPE_FUNC_SINGLE (default) 1h: HEADER_TYPE_FUNC_MULTI

Bits	Reset	R/W	Description
22:16	0	R	T_XUSB_DEV_CFG_3_HEADER_TYPE_DEVICE: The HEADER_TYPE bits identify the layout of the bytes 10h through 3Fh in configuration space and also whether or not the device contains multiple functions. Bit 7 in this register is used to identify a multi-function device. If the bit is 0, then the device is single function. If the bit is 1, then the device has multiple functions. Bits 6 through 0 specify the layout of bytes 10h through 3Fh. The LATENCY_TIMER and HEADER_TYPE are defined by parameters per block. 0h: HEADER_TYPE_DEVICE_NON_BRIDGE (default) 1h: HEADER_TYPE_DEVICE_P2P_BRIDGE
15:11	0	R	T_XUSB_DEV_CFG_3_LATENCY_TIMER: The LATENCY_TIMER bits contain, in units of PCI bus clocks, the value of the Latency Timer for this PCI bus master. This register must be implemented as writable by any master that can burst more than two data phases. This register may be implemented as read-only for devices that burst two or fewer data phases, but the hardwired value must be limited to 16 or less. A typical implementation would be to build the five high-order bits (leaving the bottom three as read-only), resulting in a timer granularity of eight clocks. At reset, the register should be set to 0 (if programmable). LATENCY_TIMER bits are writable. 0h: LATENCY_TIMER_0_CLOCKS (default) 1h: LATENCY_TIMER_8_CLOCKS 1Eh: LATENCY_TIMER_240_CLOCKS 1Fh: LATENCY_TIMER_248_CLOCKS
10:8	0	R	Reserved
7:0	0	R	T_XUSB_DEV_CFG_3_CACHE_LINE_SIZE: 0h: CACHE_LINE_SIZE_0 (default) 20h: CACHE_LINE_SIZE_32 40h: CACHE_LINE_SIZE_64

22.16.14.5 T_XUSB_DEV_CFG_4

PCI Configuration Register

Offset: 0x10 | Read/Write: R/W

Bits	Reset	R/W	Description
31:15	0	R/W	T_XUSB_DEV_CFG_4_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device required by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all don't-care address bits, effectively specifying the address space required. 0h: BASE_ADDRESS_DEFAULT (default)
14:4	0	R	T_XUSB_DEV_CFG_4_BAR_SIZE_32KB: 0h: BAR_SIZE_32KB_RSVD (default)
3	1h	R	T_XUSB_DEV_CFG_4_PREFETCHABLE: 0h: PREFETCHABLE_NOT 1h: PREFETCHABLE_MERGABLE (default)
2:1	2h	R	T_XUSB_DEV_CFG_4_ADDRESS_TYPE: The ADDRESS_TYPE bits contain the type of the Base Address. It can be 32 bits, 20 bits, or 64 bits wide. 0h: ADDRESS_TYPE_32_BIT 2h: ADDRESS_TYPE_64_BIT (default)
0	0	R	T_XUSB_DEV_CFG_4_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (default) 1h: SPACE_TYPE_IO

22.16.14.6 T_XUSB_DEV_CFG_5

Offset: 0x14 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_CFG_5_BASE_ADDRESSHI: 0h: BASE_ADDRESSHI_DEFAULT (default)

22.16.14.7 T_XUSB_DEV_CFG_6

Offset: 0x18-0x28 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_DEV_CFG_6_RSVD: 0h: RSVD_00 (default)

22.16.14.8 T_XUSB_DEV_CFG_11

PCI Configuration Register

The SUBSYSTEM_VENDOR_ID bits and SUBSYSTEM_ID bits are used to uniquely identify the add-in board or subsystem where the device resides. When the device is on the motherboard, there is no serial ROM and the registers both initialize to NONE. The motherboard BIOS must set the values of the Subsystem ID and Subsystem Vendor ID by writing the proper values to the SUBSYSTEM_VENDOR_ID and SUBSYSTEM_ID bits in the PCI_T_16 register (NOT PCI_T_11).

Offset: 0x2c | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_DEV_CFG_11_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R	T_XUSB_DEV_CFG_11_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

22.16.14.9 T_XUSB_DEV_CFG_12

PCI Configuration Register

Offset: 0x30 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_DEV_CFG_12_RESERVED: 0h: RESERVED_0 (default)

22.16.14.10 T_XUSB_CFG_13

PCI Capability Pointer Register

Offset: 0x34 | Read/Write: R

Bits	Reset	R/W	Description
31:8	0	R	Reserved
7:0	44h	R	T_XUSB_DEV_CFG_13_CAP_PTR: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. This always points to 0x44 where at least the PCI-PM registers are expected to reside. 44h: CAP_PTR_PMCAP C0h: CAP_PTR_MSI 70h: CAP_PTR_MSIX DCh: CAP_PTR_MMAP 44h: CAP_PTR_DEFAULT (default)

22.16.14.11 T_XUSB_DEV_CFG_14

PCI Configuration Register

Offset: 0x38 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_DEV_CFG_14_RESERVED: 0h: RESERVED_0 (default)

22.16.14.12 T_XUSB_DEV_CFG_15

PCI Configuration Register

Offset: 0x3c | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_DEV_CFG_15_MAX_LAT: The MAX_LAT bits contain the maximum time the device requires to gain access to the CPI bus. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microseconds. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MAX_LAT is nonzero. The INTR_PIN, MIN_GNT, and MAX_LAT are configurable per block. For a 64-byte buffer with two 32-byte sections, the maximum tolerable latency for the XHCI once a large XUSB ISO transaction has begun is about 23 µs. The latency timer is hardwired to 20 µs. This value only applies in External PCI operation. 0h: MAX_LAT_NO_REQUIREMENTS (default) 14h: MAX_LAT_5US 50h: MAX_LAT_20US
23:16	0	R	T_XUSB_DEV_CFG_15_MIN_GNT: The MIN_GNT bits contain the length of the burst period a device needs assuming a clock rate of 33 MHz. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MIN_GNT is nonzero. 0h: MIN_GNT_NO_REQUIREMENTS (default) 1h: MIN_GNT_240NS
15:8	1h	R	T_XUSB_DEV_CFG_15_INTR_PIN: The INTR_PIN bits contain the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that do not use an interrupt pin must put a 0 in this register. 0h: INTR_PIN_NONE 1h: INTR_PIN_INTA (default) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	0	R/W	T_XUSB_DEV_CFG_15_INTR_LINE: The INTR_LINE bits contain the interrupt routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this register as it initializes and configures the system. The value in this register tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is initialized to 0xff (no connection) at reset. Some PCI BIOS' cannot handle aliased INTR_LINEs. Some PCI BIOS' cannot handle INTR_LINE initialized to 0xff. 0h: INTR_LINE_IRQ0 (default) 1h: INTR_LINE_IRQ1 Fh: INTR_LINE_IRQ15 FFh: INTR_LINE_UNKNOWN

22.16.14.13 T_XUSB_DEV_CFG_16

Backdoor register write for updating subsystem ID/vendor ID.

Offset: 0x40 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_DEV_CFG_16_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R/W	T_XUSB_DEV_CFG_16_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

22.16.14.14 T_XUSB_DEV_CFG_17

Offset: 0x44 | Read/Write: R

Bits	Reset	R/W	Description
31	1h	R	T_XUSB_DEV_CFG_17_D3CPME_SUPPORT: 1h: D3CPME_SUPPORT_YES 0h: D3CPME_SUPPORT_NO
30	1h	R	T_XUSB_DEV_CFG_17_D3HPME_SUPPORT: 1h: D3HPME_SUPPORT_YES 0h: D3HPME_SUPPORT_NO
29	0	R	T_XUSB_DEV_CFG_17_D2PME_SUPPORT: 1h: D2PME_SUPPORT_YES 0h: D2PME_SUPPORT_NO
28	0	R	T_XUSB_DEV_CFG_17_D1PME_SUPPORT: 1h: D1PME_SUPPORT_YES 0h: D1PME_SUPPORT_NO
27	1h	R	T_XUSB_DEV_CFG_17_D0PME_SUPPORT: 1h: D0PME_SUPPORT_YES 0h: D0PME_SUPPORT_NO
26	0	R	T_XUSB_DEV_CFG_17_D2_SUPPORT: 0h: D2_SUPPORT_NO 1h: D2_SUPPORT_YES
25	0	R	T_XUSB_DEV_CFG_17_D1_SUPPORT: 0h: D1_SUPPORT_NO 1h: D1_SUPPORT_YES
24:22	0	R	T_XUSB_DEV_CFG_17_AUXCUR: 0h: AUXCUR_SELF 1h: AUXCUR_55MA 2h: AUXCUR_100MA 3h: AUXCUR_160MA 4h: AUXCUR_220MA 5h: AUXCUR_270MA 6h: AUXCUR_320MA 7h: AUXCUR_375MA
21	0	R	T_XUSB_DEV_CFG_17_DSI: 0h: DSI_NONE 1h: DSI_NEEDED
20	0	R	T_XUSB_DEV_CFG_17_RSVD: 0h: RSVD_0
19	0	R	T_XUSB_DEV_CFG_17_PMECLK: 0h: PMECLK_NOT_REQUIRED 1h: PMECLK_REQUIRED
18:16	3h	R	T_XUSB_DEV_CFG_17_VER: 3h: VER_1P2
15:8	C0h	R	T_XUSB_DEV_CFG_17_NEXT_PTR: C0h: NEXT_PTR_MSI 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 0h: NEXT_PTR_NULL C0h: NEXT_PTR_DEFAULT (default)
7:0	1h	R	T_XUSB_DEV_CFG_17_CAP: 1h: CAP_PCIPM

22.16.14.15 T_XUSB_DEV_CFG_18_PMCSR

Offset: 0x48 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23:16	0	R	T_XUSB_DEV_CFG_18_PMCSR_BSE_RSVD: 0h: BSE_RSVD_00

Bits	Reset	R/W	Description
15	0	RW1C	T_XUSB_DEV_CFG_18_PMCSR_PMESTATUS: 0h: PMESTATUS_NOT_PENDING (default) 1h: PMESTATUS_PENDING 1h: PMESTATUS_CLEAR
14:13	0	R	T_XUSB_DEV_CFG_18_PMCSR_DSCALE: 0h: DSCALE_INIT
12:9	0	R	T_XUSB_DEV_CFG_18_PMCSR_DSEL: 0h: DSEL_INIT
8	0	R/W	T_XUSB_DEV_CFG_18_PMCSR_PME: 1h: PME_ENABLE 0h: PME_DISABLE (default)
7:4	0	R	T_XUSB_DEV_CFG_18_PMCSR_RSVD1: 0h: RSVD1_00
3	1h	R	T_XUSB_DEV_CFG_18_PMCSR_NSR: 1h: NSR_NORESET 0h: NSR_RESET
2	0	R	T_XUSB_DEV_CFG_18_PMCSR_RSVD0: 0h: RSVD0_0
1:0	0	R/W	T_XUSB_DEV_CFG_18_PMCSR_PWRSTATE: 0h: PWRSTATE_D0 (default) 1h: PWRSTATE_D1 2h: PWRSTATE_D2 3h: PWRSTATE_D3H

22.16.14.16 T_XUSB_DEV_MSI_CTRL

MSI Message Control and Capability Register B0h

MSI_CTRL is the main control register for MSI support.

Offset: 0xc0 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	Reserved
24	0	R	T_XUSB_DEV_MSI_CTRL_VECTOR_MASK_CAP: The VECTOR_MASK_CAP field indicates whether or not the controller supports MSI-per-vector masking. 0h: VECTOR_MASK_CAP_DIS 1h: VECTOR_MASK_CAP_EN 0h: VECTOR_MASK_CAP_DEFAULT (default)
23	1h	R	T_XUSB_MSI_CTRL_64_ADDR_CAP: The 64_ADDR_CAP field indicates whether or not the controller is capable of generating a 64-bit message address. A value of 1 means the controller is capable of generating a 64-bit message address. 0h: 64_ADDR_CAP_DIS 1h: 64_ADDR_CAP_EN 1h: 64_ADDR_CAP_DEFAULT (default)
22:20	0	R/W	T_XUSB_DEV_MSI_CTRL_MULT_MSG_ENABLE: System software writes to this field to indicate the number of allocated vectors (less than or equal to the number of vectors requested). The number of vectors is aligned as a power of two. When MSI is enabled, the controller will be allocated at least one vector. 0h: MULT_MSG_ENABLE_1 1h: MULT_MSG_ENABLE_2 2h: MULT_MSG_ENABLE_4 3h: MULT_MSG_ENABLE_8 4h: MULT_MSG_ENABLE_16 5h: MULT_MSG_ENABLE_32 0h: MULT_MSG_ENABLE_DEFAULT (default)

Bits	Reset	R/W	Description
19:17	0	R	T_XUSB_DEV_MSI_CTRL_MULT_MSG_CAP: System software reads this field to determine the number of requested vectors. The number of requested vectors must be aligned to a power of two. Values of 6 and 7 in this field are reserved. 0h: MULT_MSG_CAP_1 1h: MULT_MSG_CAP_2 2h: MULT_MSG_CAP_4 3h: MULT_MSG_CAP_8 4h: MULT_MSG_CAP_16 5h: MULT_MSG_CAP_32 0h: MULT_MSG_CAP_DEFAULT (default)
16	0	R/W	T_XUSB_DEV_MSI_CTRL_MSI_ENABLE: The MSI_ENABLE field enables the MSI capability. If MSI_ENABLE is written to a 1, the controller is permitted to use MSI to request service and is prohibited from using the legacy interrupt. System configuration software sets this bit to enable MSI. A device driver is prohibited from writing this bit to mask the controller's service request. If this bit is written to a 0, the controller is prohibited from using MSI to request service. 0h: MSI_ENABLE_OFF 1h: MSI_ENABLE_ON 0h: MSI_ENABLE_DEFAULT (default)
15:8	E0h	R	T_XUSB_DEV_MSI_CTRL_NEXT_PTR: The NEXT_PTR field identifies the next item in the capabilities list. It is a read-only field. 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 44h: NEXT_PTR_PMCAP 0h: NEXT_PTR_NULL 0h: NEXT_PTR_DEFAULT (default)
7:0	5h	R	T_XUSB_DEV_MSI_CTRL_CAP_ID: The CAP_ID field identifies this capability block as the MSI capability block. This is read-only as 0x5. 5h: CAP_ID_MSI (default)

22.16.14.17 T_XUSB_DEV_MSI_ADDR1

MSI Message Address Register 84h

MSI_ADDR1 specifies the lower 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc4 | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R/W	T_XUSB_DEV_MSI_ADDR1_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the Dword-aligned address for the MSI memory write transaction. 0h: MSG_ADDR_DEFAULT (default)
1:0	0	R	Reserved

22.16.14.18 T_XUSB_DEV_MSI_ADDR2

MSI Message Upper Address Register 88h

MSI_ADDR2 specifies the upper 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc8 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_MSI_ADDR2_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the upper 32 bits of the address for the MSI memory write transaction. The contents of this register only apply when T_XUSB_CFG_MSI_CTRL_64_ADDR_CAP bit is set. 0h: MSG_ADDR_DEFAULT (default)

22.16.14.19 T_XUSB_DEV_MSI_DATA

MSI Message Data Register 8Ch

The MSI_DATA register contains the system-specified message.

Offset: 0xcc | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	0	R/W	T_XUSB_DEV_MSI_DATA_MSG_DATA: System-specified message When MSI is enabled, the message data is driven onto the lower 16 bits of the MSI memory write. The MULT_MSG_ENABLE field in configuration register 80h specifies the number of low-order message data bits that the XHCI is permitted to modify to generate its system software allocated vectors. 0h: MSG_DATA_DEFAULT (default)

22.16.14.20 T_XUSB_DEV_MSI_MASK

MSI Message Data Register 60h

The MSI_MASK register enables software to mask or defer message generation on a per-vector basis.

Offset: 0xD0 | Read/Write: R/W

Bits	Reset	R/W	Description
31:1	0	R/W	Reserved
0	0	R/W	XUSB_DEV_MSI_MASK_BIT: 0h: MSI_MASK_BIT_DEFAULT (default)

22.16.14.21 XUSB_DEV_MSI_PEND

MSI Pending Bits Register 64h

The MSI_PEND register shows which MSI vectors have pending interrupts.

Offset: 0xD4 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	0	R	XUSB_DEV_MSI_MASK_BIT: 0h: MSI_PEND_BIT_DEFAULT (default)

22.16.15 XUSB Device Controller Registers

Base Address: XUSB_DEV_BASE

22.16.15.1 T_XUSB_DEV_XHCI_SPARAM

ID Register

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:21	0	R	Reserved
20:16	2h	R	XUSB_DEV_XHCI_SPARAM_ERSTMAX: Event Ring Segment Table Max Indicates the maximum value supported in the Event Ring Segment Table Base Size, where the maximum size of the Event Ring Segment Table is 2ERST Max 2h: SPARAM_ERSTMAX_VALUE

Bits	Reset	R/W	Description
15:0	1h	R	T_XUSB_DEV_XHCI_SPARAM_RSVD0_INIT: 1h: SPARAM_RSVD0_INIT

22.16.15.2 T_XUSB_DEV_XHCI_DB

Doorbell Register

Offset: 0x04 | Read/Write: W

Bits	Reset	R/W	Description
31:16	0	W	T_XUSB_DEV_XHCI_DB_STREAMID: Only significant for control EPs and is used to represent the control sequence number. Software is responsible for ensuring that the value written to this field matches the Sequence Number of a SETUP Packet Event STREAMID is not used for stream EPs. 0h: STREAMID_00 (default)
15:8	0	W	T_XUSB_DEV_XHCI_DB_TARGET: The target field represents the Endpoint ID to which the doorbell is targeted 00h: EP 0 Enqueue Pointer Update (Control EP 0) even values: OUT Enqueue Pointer Update odd values: IN Enqueue Pointer Update 0h: TARGET_INIT
7:0	0	R	T_XUSB_DEV_XHCI_DB_RSVD0: 0h: RSVD0_00 (default)

22.16.15.3 T_XUSB_DEV_XHCI_ERSTSZ

Event Ring Segment Table Size Register

Specify the size of each event ring segment in terms of number of event TRB slots (each slot is 16 bytes) the min value of size in each register is 16 and the max is 4096 both the segments should be initialized with valid values

Offset: 0x08 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_DEV_XHCI_ERSTSZ_ERST1SZ: 0h: ERST1SZ_INIT(default)
15:0	0	R/W	T_XUSB_DEV_XHCI_ERSTSZ_ERST0SZ: 0h: ERST0SZ_INIT (default)

22.16.15.4 T_XUSB_DEV_XHCI_RSVD0

Offset: 0x0C | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_DEV_XHCI_ERSTSZ_ERST1SZ: 0h: RSVD0_RSVD0_00 (default)

22.16.15.5 T_XUSB_DEV_XHCI_ERST0BALO

Event Ring Segment 0 Base Address Register

Offset: 0x10 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_DEV_XHCI_ERST0BALO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_DEV_XHCI_ERST0BALO_RSVD0: 0h: RSVD0_00 (default)

22.16.15.6 T_XUSB_DEV_XHCI_ERST0BAHI

Offset: 0x14 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_ERST0BAHI_ADDRHI: 0h: ADDRHI_INIT (default)

22.16.15.7 T_XUSB_DEV_XHCI_ERST1BALO

Event Ring Segment 1 Base Address Register

Offset: 0x18 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_DEV_XHCI_ERST1BALO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_DEV_XHCI_ERST1BALO_RSVD0: 0h: RSVD0_00 (default)

22.16.15.8 T_XUSB_DEV_XHCI_ERST1BAHI

Offset: 0x1C | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_ERST1BAHI_ADDRHI: 0h: ADDRHI_INIT (default)

22.16.15.9 T_XUSB_DEV_XHCI_ERDPLO

Event Ring Dequeue Pointer Register

Reflects the current value of ERDP, software updates this register after it has popped events

Offset: 0x20 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_DEV_XHCI_ERDPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3	0	RW1C	T_XUSB_DEV_XHCI_ERDPLO_EHB: EHB is set by hardware when an event is posted and is cleared by software after it has processed all events. 0h: EHB_INIT (default) 1h: EHB_CLEAR
2:0	0	R	T_XUSB_DEV_XHCI_ERDPLO_RSVD0: 0h: RSVD0_00 (default)

22.16.15.10 T_XUSB_DEV_XHCI_ERDPHI

Offset: 0x24 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_ERDPHI_ADDRHI: 0h: ADDRHI_INIT (default)

22.16.15.11 T_XUSB_DEV_XHCI_EREPLO

Event Ring Enqueue Pointer

Reflects the current value of Event Ring Enqueue Pointer; it is updated by hardware after posting an event software can write to the register when Control Enable bit is cleared. Software specifies the segment index to be used and hardware samples the value written by software on a positive edge of control enable.

Offset: 0x28 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_DEV_XHCI_EREPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3:2	0	R	T_XUSB_DEV_XHCI_EREPLO_RSVD0: 0h: RSVD0_00
1	0	R/W	T_XUSB_DEV_XHCI_EREPLO_SEGI: 0h: SEGI_INIT (default) 1h: SEGI_MAX
0	0	R/W	T_XUSB_DEV_XHCI_EREPLO_ECS: 0h: ECS_INIT (default)

22.16.15.12 T_XUSB_DEV_XHCI_EREPHI

Offset: 0x2C | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_EREPHI_ADDRHI: 0h: ADDRHI_INIT (default)

22.16.15.13 T_XUSB_DEV_XHCI_CTRL

Control Register

Offset: 0x30 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R/W	T_XUSB_DEV_XHCI_CTRL_ENABLE: Device control Enable used to enable Device Mode operation 0h: ENABLE_DIS 1h: ENABLE_EN
30:24	0	R/W	T_XUSB_DEV_XHCI_CTRL_DEVADR: Address assigned to the device DUT 0h: DEVADR_INIT
23:8	0	R	T_XUSB_DEV_XHCI_CTRL_RSVD0: 0h: RSVD0_00
7	0	R/W	T_XUSB_DEV_XHCI_CTRL_EWE: Enable event for MFINDEX rollover from 3FFF to 0 0h: EWE_FALSE (default) 1h: EWE_TRUE
6	0	R/W	T_XUSB_DEV_XHCI_CTRL_SMI_DSE: Enable SMI interrupt for device system errors 0h: SMI_DSE_FALSE (default) 1h: SMI_DSE_TRUE
5	0	R/W	XUSB_DEV_XHCI_CTRL_SMI_EVT: 0h: EVT_FALSE (default) 1h: EVT_TRUE
4	0	R/W	XUSB_DEV_XHCI_CTRL_IE: Enable legacy/smi interrupt for pending events 0h: IE_FALSE (default) 1h: IE_TRUE
3:2	0	R	XUSB_DEV_XHCI_CTRL_RSVD1: 0h: RSVD1_00

Bits	Reset	R/W	Description
1	0	R/W	T_XUSB_DEV_XHCI_CTRL_LSE: Generate Event on PORTSC change 0h: LSE_DIS (default) 1h: LSE_EN
0	0	R/W	T_XUSB_DEV_XHCI_CTRL_RUN: Device Mode Run/Stop bit 0h: RUN_STOP (default) 1h: RUN_RUN

22.16.15.14 T_XUSB_DEV_XHCI_ST

Status Register

Run Change - set when run bit transitions from 1 to 0

Interrupt Pending - set by hardware when an event is added to the event ring

Device System Error bit set when error is received on FPCI wrapper

Offset: 0x34 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	R	T_XUSB_DEV_XHCI_ST_RSVD1: 0h: RSVD1_00 (default)
5	0	RW1C	T_XUSB_DEV_XHCI_ST_DSE: 1h: DSE_PENDING (default) 1h: DSE_CLEAR
4	0	RW1C	T_XUSB_DEV_XHCI_ST_IP: 0h: IP_NOTPENDING (default) 1: IP_PENDING 1h: IP_CLEAR
3:1	0	R	T_XUSB_DEV_XHCI_ST_RSVD0: 0h: RSVD0_00
0	0	RW1C	T_XUSB_DEV_XHCI_ST_RC: 0h: RC_NOTPENDING 1h: RC_PENDING 1h: RC_CLEAR

22.16.15.15 T_XUSB_DEV_XHCI_RT_IMOD

Offset: 0x38 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	FA0h	R/W	T_XUSB_DEV_XHCI_RT_IMOD_IMODC: FA0h: IMODC_INIT
15:0	FA0h	R/W	T_XUSB_DEV_XHCI_RT_IMOD_IMODI: FA0h: IMODI_INIT

22.16.15.16 T_XUSB_DEV_XHCI_PORTSC

Offset: 0x3C | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	Reserved
30	0	R	T_XUSB_DEV_XHCI_PORTSC_WPR: 0h: WPR_NORST (default) 1h: WPR_RST
29:28	0	R	T_XUSB_DEV_XHCI_PORTSC_RSVD4: 0h: RSVD4_00

Bits	Reset	R/W	Description
27:24	0	R	Reserved
23	0	RW1C	T_XUSB_DEV_XHCI_PORTSC_CEC: 0h: CEC_NOT_PENDING (default) 1h: CEC_PENDING 1h: CEC_CLEAR
22	0	RW1C	T_XUSB_DEV_XHCI_PORTSC_PLC: 0h: PLC_NOT_PENDING 1h: PLC_ PENDING 1h: PLC_CLEAR
21	0	RW1C	T_XUSB_DEV_XHCI_PORTSC_PRC: 0h: PRC_NOT_PENDING 1h: PRC_ PENDING 1h: PRC_CLEAR
20	0	R	Reserved
19	0	RW1C	T_XUSB_DEV_XHCI_PORTSC_WRC: 0h: WRC_NOT_PENDING 1h: WRC_PENDING 1h: WRC_CLEAR
18	0	R	T_XUSB_DEV_XHCI_PORTSC_RSVD3: 0h: RSVD3_00
17	0	RW1C	T_XUSB_DEV_XHCI_PORTSC_CSC: 0h: CSC_NOT_PENDING 1h: CSC_ PENDING 1h: CSC_CLEAR
16	0	R/W	T_XUSB_DEV_XHCI_PORTSC_LWS: 0h: LWS_INIT (default) 1h: LWS_SET
15	0	R	Reserved
14	0	R	T_XUSB_DEV_XHCI_PORTSC_RSVD2: 0h: RSVD2_00
13:10	0	R	T_XUSB_DEV_XHCI_PORTSC_PS: 0h: PS_UNDEFINED (default) 1h: PS_FS 2h: PS_LS 3h: PS_HS 4h: PS_SS
9	0	R	T_XUSB_DEV_XHCI_PORTSC_LANE_POLARITY_VALUE: 0h: POLARITY_VALUE_INIT
8:5	0	R/W	T_XUSB_DEV_XHCI_PORTSC_PLS: 0h: PLS_U0 1h: PLS_U1 2h: PLS_U2 3h: PLS_U3 4h: PLS_DISABLED 5h: PLS_RXDETECT 6h: PLS_INACTIVE 7h: PLS_POLLING 8h: PLS_RECOVERY 9h: PLS_HOTRESET Ah: PLS_COMPLIANCE Bh: PLS_LOOPBACK Fh: PLS_RESUME
4	0	R	T_XUSB_DEV_XHCI_PORTSC_PR: 0h: PR_NORST 1h: PR_RST
3	0	R/W	T_USB_DEV_XHCI_PORTSC_LANE_POLARITY_OVRD_VALUE: 0h: VALUE_INIT
2	0	R/W	T_XUSB_DEV_XHCI_PORTSC_LANE_POLARITY_OVRD 0h: OVRD_INIT

Bits	Reset	R/W	Description
1	0	R/W	T_XUSB_DEV_XHCI_PORTSC_PED: 0h: PED_DIS 1h: PED_EN
0	0	R/W	T_XUSB_DEV_XHCI_PORTSC_CCS: 0h: CS_NOCON 1h: CCS_CON

22.16.15.17 T_XUSB_DEV_XHCI_ECPLO

Offset: 0x40 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	R	T_XUSB_DEV_XHCI_ECPLO_ADDRLO: 0h: ADDRLO_INIT
5:0	0	R/W	T_XUSB_DEV_XHCI_ECPLO_RSVD0: 0h: RSVD0_00

22.16.15.18 T_XUSB_DEV_XHCI_ECPHI

Offset: 0x44 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_ECPHI_ADDRHI: 0h: ADDRHI_INIT

22.16.15.19 T_XUSB_DEV_XHCI_MFINDEX

Offset: 0x48 | Read/Write: R

Bits	Reset	R/W	Description
31:14	None	R	Reserved
13:3	None	R	T_XUSB_DEV_XHCI_MFINDEX_FRAME: 0h: FRAME_INIT
2:0	0	R	T_XUSB_DEV_XHCI_MFINDEX_UFRAME: UFRAME_INIT

22.16.15.20 T_XUSB_DEV_XHCI_PORTPM

Port PM Status and Control Register

Offset: 0x4C | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R/W	T_XUSB_DEV_XHCI_PORTPM_PNG_CYA: 0h: PNG_CYA_INIT (default)
30	0	R/W	T_XUSB_DEV_XHCI_PORTPM_FRWE: 0h: FRWE_INIT (default)
29	0	R/W	T_XUSB_DEV_XHCI_PORTPM_U2E: 0h: U2E_INIT (default)
28	0	R/W	T_XUSB_DEV_XHCI_PORTPM_U1E: 0h: U1E_INIT (default)
27	1h	R/W	T_XUSB_DEV_XHCI_PORTPM_WOD: 1h: WOD_INIT (default)
26	1h	R/W	T_XUSB_DEV_XHCI_PORTPM_WOC: 1h: WOC_INIT (default)
25	unknown	R	T_XUSB_DEV_XHCI_PORTPM_VBA: 0h: VBA_INIT

Bits	Reset	R/W	Description
24	0	R/W	T_XUSB_DEV_XHCI_PORTPM_FLA: 0h: FLA_INIT (default)
23:16	FFh	R/W	T_XUSB_DEV_XHCI_PORTPM_U1TIMEOUT: FFh: U1TIMEOUT_INIT (default)
15:8	FFh	R/W	T_XUSB_DEV_XHCI_PORTPM_U2TIMEOUT: FFh: U2TIMEOUT_INIT (default)
7:4	0	R/W	T_XUSB_DEV_XHCI_PORTPM_HIRD: 0h: HIRD_INIT (default)
3	0	R/W	T_XUSB_DEV_XHCI_PORTPM_RWE: 0h: RWE_DISABLED (default) 1h: RWE_ENABLED
2	0	R	Reserved
1:0	1h	R/W	T_XUSB_DEV_XHCI_PORTPM_L1S: 0h: L1S_DROP 1h: L1S_ACCEPT 2h: L1S_NYET 3h:L1S_STALL

22.16.15.21 T_XUSB_DEV_XHCI_EP_HALT

EP HALT register used by software to program an EP to respond to host request with STALL transaction

Offset: 0x50 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_EP_HALT_DCI: 0h: DCI_NO (default) 1h: DCI_YES

22.16.15.22 T_XUSB_DEV_XHCI_EP_PAUSE

EP Pause register used by software to program the DUT to stop processing the EP and respond host requests with a NRDY/ NAK

Offset: 0x54 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_EP_PAUSE_DCI: 0h: DCI_NO (default) 1h: DCI_YES

22.16.15.23 T_XUSB_DEV_XHCI_EP_RELOAD

EP Reload bits are used by software to direct the Device BI to reload the EP context for that EP and update its internal EP related variables like EP state.

Offset: 0x58 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_EP_RELOAD_DCI: 0h: DCI_NO (default) 1h: DCI_YES

22.16.15.24 T_XUSB_DEV_XHCI_EP_STCHG

EP STCHG is set by hardware whenever halt/pause/run bit is cleared by software, to indicate completion of processing of the software request.

Offset: 0x5C | Read/Write: R

Bits	Reset	R/W	Description
31	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN15 0h: IN15_NO 1h:IN15_YES 1h: IN15_CLEAR
30	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT15: 0h: OUT15_NO 1h: OUT15_YES 1h: OUT15_CLEAR
29	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN14: 0h: IN14_NO 1h:IN14_YES 1h: IN14_CLEAR
28	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT14: 0h: OUT14_NO 1h: OUT14_YES 1h: OUT14_CLEAR
27	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN13: 0h: IN13_NO 1h:IN13_YES 1h: IN13_CLEAR
26	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT13: 0h: OUT13_NO 1h: OUT13_YES 1h: OUT13_CLEAR
25	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN12: 0h: IN12_NO 1h: IN12_YES 1h: IN12_CLEAR
24	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT12: 0h: OUT12_NO 1h: OUT12_YES 1h: OUT12_CLEAR
23	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN11: 0h: OUT11_NO 1h: OUT11_YES 1h: OUT11_CLEAR
22	0	RW1C	XUSB_DEV_XHCI_EP_STCHG_OUT11: 0h: OUT11_NO 1h: OUT11_YES 1h: OUT11_CLEAR
21	0	RW1C	XUSB_DEV_XHCI_EP_STCHG_IN10: 0h: IN10_NO 1h:OUT10_YES 1h: OUT10_CLEAR
20	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT10: OUT10_NO OUT10_YES OUT10_CLEAR
19	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN9: 0h: IN9_NO 1h: IN9_YES 1h: IN9_CLEAR

Bits	Reset	R/W	Description
18	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT9: 0h: OUT9_NO 1h: OUT9_YES 1h: OUT9_CLEAR
17	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN8: 0h: IN8_NO 1h: IN8_YES 1h: IN8_CLEAR
16	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT8: 0h: OUT8_NO 1h: OUT8_YES 1h: OUT8_CLEAR
15	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN7: 0h: IN7_NO 1h: IN7_YES 1h: IN7_CLEAR
14	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT7: 0h: OUT7_NO 1h: OUT7_YES 1h: OUT7_CLEAR
13	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN6: 0h: IN6_NO 1h: IN6_YES 1h: IN6_CLEAR
12	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT6: 0h: OUT6_NO 1h: OUT6_YES 1h: OUT6_CLEAR
11	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN5: 0h: IN5_NO 1h: IN5_YES 1h: IN5_CLEAR
10	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT5: 0h: OUT5_NO 1h: OUT5_YES 1h: OUT5_CLEAR
9	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN4: 0h: IN4_NO 1h: IN4_YES 1h: IN4_CLEAR
8	0	RW1C	XUSB_DEV_XHCI_EP_STCHG_OUT4: 0h: OUT4_NO 1h: OUT4_YES 1h: OUT4_CLEAR
7	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN3: 0h: IN3_NO 1h: IN3_YES 1h: IN3_CLEAR
6	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT3: 0h: OUT3_NO 1h: OUT3_YES 1h: OUT3_CLEAR
5	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN2: 0h: IN2_NO 1h: IN2_YES 1h: IN2_CLEAR

Bits	Reset	R/W	Description
4	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT2: 0h: OUT2_NO 1h: OUT2_YES 1h: OUT2_CLEAR
3	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN1: 0h: IN1_NO 1h: IN1_YES 1h: IN1_CLEAR
2	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT1: 0h: OUT1_NO 1h: OUT1_YES 1h: OUT1_CLEAR
1	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_IN0: 0h: IN0_NO (default) 1h: IN0_YES 1h: IN0_CLEAR
0	0	RW1C	T_XUSB_DEV_XHCI_EP_STCHG_OUT0: 0h: UT0_NO(default) 1h: OUT0_YES 1h: OUT0_CLEAR

22.16.15.25 T_XUSB_DEV_XHCI_FLOWCNTRL

Flow control threshold values for HSFS NAK - software needs to enable this if more than 4 IN or 4 OUT EPs are configured

Offset: 0x60 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_DEV_XHCI_FLOWCNTRL_RSVD0: 0h: RSVD0_00
23:16	8h	R/W	T_XUSB_DEV_XHCI_FLOWCNTRL_IDLE_MITS: 8h: IDLE_MITS_INIT
15	0	R/W	XUSB_DEV_XHCI_FLOWCNTRL_OUT_EN: 0h: OUT_EN_INIT
14:8	0	R/W	XUSB_DEV_XHCI_FLOWCNTRL_OUT_THRESH: 0h: OUT_THRESH_INIT
7	0	R/W	XUSB_DEV_XHCI_FLOWCNTRL_IN_EN: 0h: IN_EN_INIT
6:0	0	R/W	T_XUSB_DEV_XHCI_FLOWCNTRL_IN_THRESH: 40h: IN_THRESH_INIT

22.16.15.26 T_XUSB_DEV_XHCI_DEVNOTIF_LO

Device notification registers, for software to direct hardware to send a device notification TP data should be written and updated before writing 1 to the trigger.

Offset: 0x64 | Read/Write: R/W

Bits	Reset	R/W	Description
31:8	0	R/W	T_XUSB_DEV_XHCI_DEVNOTIF_LO_DATA: 0h: DATA_INIT (default)
7:4	0	R/W	T_XUSB_DEV_XHCI_DEVNOTIF_LO_TYPE: 0h: TYPE_INIT (default)
3:1	0	R	T_XUSB_DEV_XHCI_DEVNOTIF_LO_RSVD0: 0h: LO_RSVD0

Bits	Reset	R/W	Description
0	0	R/W	T_XUSB_DEV_XHCI_DEVNOTIF_LO: 0h: TRIG_INIT (default) 1h: TRIG_SET

22.16.15.27 T_XUSB_DEV_XHCI_DEVNOTIF_HI

Offset: 0x68 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_DEV_XHCI_DEVNOTIF_HI_DATA: 0h: DATA_INIT

22.16.15.28 T_XUSB_DEV_XHCI_PORTHALT

Offset: 0x6C | Read/Write: R/W

Bits	Reset	R/W	Description
31:27	0	R	Reserved
26	0	R/W	T_XUSB_DEV_XHCI_PORTHALT_STCHG_REQ: 0h: EN_INIT (default)
25	0	R/W	T_XUSB_DEV_XHCI_PORTHALT_STCHG_PME_EN: 0h: EN_INIT (default)
24	1h	R/W	T_XUSB_DEV_XHCI_PORTHALT_STCHG_INTR_EN: 1h: EN_INIT (default)
23:21	0	R	Reserved
20	0	RW1C	T_XUSB_DEV_XHCI_PORTHALT_STCHG_REQ: 0h: REQ_NOT_PENDING (default) 1h: REQ_PENDING 1h: REQ_CLEAR
19:16	0	R	T_XUSB_DEV_XHCI_PORTHALT_STCHG_STATE: 0h: STATE_U0 (default)
15:2	0	R	Reserved
1	0	RW1C	T_XUSB_DEV_XHCI_PORTHALT_HALT_REJECT: 0h: REJECT_FALSE (default) 1h: REJECT_TRUE 1h: REJECT_CLEAR
0	1h	R/W	T_XUSB_DEV_XHCI_PORTHALT_HALT_LTSSM: 0h: LTSSM_INIT (default)

22.16.15.29 T_XUSB_DEV_XHCI_PORT_TM

Testmode control register, for enabling DUT to enter testmode and send specific patterns

Offset: 0x70 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R	Reserved
3:0	0	R/W	T_XUSB_DEV_XHCI_PORT_TM_CTRL: 0h: CTRL_DISABLED (default) 1h: CTRL_TESTJ 2h: CTRL_TESTK 3h: CTRL_SE0_NAK 4h: CTRL_TEST_PKT 5h: CTRL_TEST_FORCEEN

22.16.15.30 T_XUSB_DEV_XHCI_EP_THREAD_ACTIVE

Per EP bit to indicate whether EP is loaded/active in BI

Offset: 0x74 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_DEV_XHCI_EP_THREAD_ACTIVE_DCI: 0h: DCI_NO (default) 1h: DCI_YES

22.16.15.31 T_XUSB_DEV_XHCI_EP_STOPPED

EP_STOPPED register

Offset: 0x78 | Read/Write: R

Bits	Reset	R/W	Description
31	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN15: 0h: IN15_NO (default) 1h: IN15_YES 1h: IN15_CLEAR
30	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT15: 0h: OUT15_NO (default) 1h: OUT15_YES 1h: OUT15_CLEAR
29	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN14: 0h: IN14_NO (default) 1h: IN14_YES 1h: IN14_CLEAR
28	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT14: 0h: OUT14_NO (default) 1h: OUT14_YES 1h: OUT14_CLEAR
27	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN13: 0h: IN13:_NO (default) 1h: IN13_YES 1h: IN13_CLEAR
26	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT13: 0h: OUT13_NO (default) 1h: OUT13_YES 1h: OUT13_CLEAR
25	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN12: 0h: IN12_NO (default) 1h: IN12_YES 1h: IN12_CLEAR
24	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT12: 0h: OUT12_NO (default) 1h: OUT12_YES 1h: OUT12_CLEAR
23	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN11: 0h: IN11_NO (default) 1h: IN11_YES 1h: IN11_CLEAR
22	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT11: 0h: OUT11_NO (default) 1h: OUT11_YES 1h: OUT11_CLEAR

Bits	Reset	R/W	Description
21	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN10: 0h: IN10_NO (default) 1h: IN10_YES 1h: IN10_CLEAR
20	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT10: 0h: OUT10_NO (default) 1h: OUT10_YES 1h: OUT10_CLEAR
19	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN9: 0h: IN9_NO (default) 1h: IN9_YES 1h: IN9_CLEAR
18	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT9: 0h: OUT9_NO (default) 1h: OUT9_YES 1h: OUT9_CLEAR
17	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN8: 0h: IN8_NO (default) 1h: IN8_YES 1h: IN8_CLEAR
16	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT8: 0h: OUT8_NO (default) 1h: OUT8_YES 1h: OUT8_CLEAR
15	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN7: 0h: IN7_NO (default) 1h: IN7_YES 1h: IN7_CLEAR
14	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT7: 0h: OUT7_NO (default) 1h: OUT7_YES 1h: OUT7_CLEAR
13	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN6: 0h: IN6_NO (default) 1h: IN6_YES 1h: IN6_CLEAR
12	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT6: 0h: OUT6_NO (default) 1h: OUT6_YES 1h: OUT6_CLEAR
11	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN5: 0h: IN5_NO (default) 1h: IN5_YES 1h: IN5_CLEAR
10	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT5: 0h: OUT5_NO (default) 1h: OUT5_YES 1h: OUT5_CLEAR
9	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN4: 0h: IN4_NO (default) 1h: IN4_YES 1h: IN4_CLEAR
8	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT4: 0h: OUT4_NO (default) 1h: OUT4_YES 1h: OUT4_CLEAR

Bits	Reset	R/W	Description
7	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN3: 0h: IN3_NO (default) 1h: IN3_YES 1h: IN3_CLEAR
6	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT3: 0h: OUT3_NO (default) 1h: OUT3_YES 1h: OUT3_CLEAR
5	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN2: 0h: IN2_NO (default) 1h: IN2_YES 1h: IN2_CLEAR
4	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT2: 0h: OUT2_NO (default) 1h: OUT2_YES 1h: OUT2_CLEAR
3	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN1: 0h: IN1_NO (default) 1h: IN1_YES 1h: IN1_CLEAR
2	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT1: 0h: OUT1_NO (default) 1h: OUT1_YES 1h: OUT1_CLEAR
1	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_IN0: 0h: IN0_NO (default) 1h: IN0_YES 1h: IN0_CLEAR
0	0	RW1C	T_XUSB_DEV_XHCI_EP_STOPPED_OUT0: 0h: OUT0_NO (default) 1h: OUT0_YES 1h: OUT0_CLEAR

22.16.15.32 T_XUSB_DEV_XHCI_STREAMID_CFG

Offset: 0x7C | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	Unknown	R/W	T_XUSB_DEV_XHCI_STREAMID_CFG_STREAMID:
15:13	0	R	Reserved
12:8	Unknown	R/W	T_XUSB_DEV_XHCI_STREAMID_CFG_DCI:
7:1	0	R	Reserved
0	0h	R/W	T_XUSB_DEV_XHCI_STREAMID_CFG_TRIGGER: 0h TRIGGER_NOT_PENDING (default) 1h TRIGGER_PENDING 1h TRIGGER_SET

22.16.15.33 T_XUSB_DEV_XHCI_DEV_SOFTRST

SOFTRST Signal for Device Mode

Offset: 0x84 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:8	80h	R/W	T_XUSB_DEV_XHCI_DEV_SOFTRST_RSTCNT: 80h: RSTCNT_INIT (default)

Bits	Reset	R/W	Description
7:2	0	R	Reserved
1	0	R/W	T_XUSB_DEV_XHCI_DEV_SOFTRST_CYA_DMAIDLE: 0h: DMAIDLE_INIT (default)
0	0	R/W	T_XUSB_DEV_XHCI_DEV_SOFTRST_RESET: 0h: RESET_NOT_PENDING (default) 1h: RESET_PENDING 1h: RESET_SET

22.16.15.34 T_XUSB_DEV_XHCI_HSFSPI_COUNT0

Offset: 0x100 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	12Ch	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT0_FS_RESET_MIN: 12Ch: RESET_MIN_INIT (default)

22.16.15.35 T_XUSB_DEV_XHCI_HSFSPI_COUNT1

Offset: 0x104 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	3395h	R/W	T_XUSB_DEV_ XHCI_HSFSPI_COUNT0_HS_RESET_MIN: 3395h: RESET_MIN_INIT

22.16.15.36 T_XUSB_DEV_XHCI_HSFSPI_COUNT2

Offset: 0x108 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	2EEFh	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT2_HS_RESET_ST_RESET_MIN: 2EEFh: RESET_ST_MIN_INIT

22.16.15.37 T_XUSB_DEV_XHCI_HSFSPI_COUNT3

Offset: 0x10C | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	3A980h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT3_TX_CHIRP_MID: 3A980h: CHIRP_MID_INIT

22.16.15.38 T_XUSB_DEV_XHCI_HSFSPI_COUNT4

Offset: 0x110 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	12C0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT4_CHIRP_MIN: 12C0h: CHIRP_MIN_INIT

22.16.15.39 T_XUSB_DEV_XHCI_HSFSPI_COUNT5

Offset: 0x114 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	1C20h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT5_CHIRP_MAX: 1C20h: CHIRP_MAX_INIT

22.16.15.40 T_XUSB_DEV_XHCI_HSFSPI_COUNT6

Offset: 0x118 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	57ED0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT6_INACTIVITY_TIMEOUT: 57ED0h: TIMEOUT_INIT

22.16.15.41 T_XUSB_DEV_XHCI_HSFSPI_COUNT7

Offset: 0x11C | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	C8h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT7_HS_FSM_TIMEOUT: C8h: TIMEOUT_INIT

22.16.15.42 T_XUSB_DEV_XHCI_HSFSPI_COUNT8

Offset: 0x120 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	B0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT8_FS_FSM_TIMEOUT: B0h: FS_FSM_TIMEOUT_INIT

22.16.15.43 T_XUSB_DEV_XHCI_HSFSPI_COUNT9

Offset: 0x124 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	124F80h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT9_U3_RESUME_K_DURATION: 124F80h: K_DURATION_INIT

22.16.15.44 T_XUSB_DEV_XHCI_HSFSPI_COUNT10

Offset: 0x128 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	3A9Eh	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT10_U3_ENTRY_DELAY: 3A9Eh: ENTRY_DELAY_INIT

22.16.15.45 T_XUSB_DEV_XHCI_HSFSPI_COUNT11

Offset: 0x12C | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	1D4FFh	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT11_FS_RESET_ST_RESET_MIN: 1D4FFh: FS_RESET_ST_RESET_MIN_INIT (default)

22.16.15.46 T_XUSB_DEV_XHCI_HSFSPI_COUNT12

Offset: 0x130 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	3D4h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT12_U2_ENTRY_DELAY: 3D4h: U2_ENTRY_DELAY_INIT (default)

22.16.15.47 T_XUSB_DEV_XHCI_HSFSPI_COUNT13

Offset: 0x134 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29:0	1772h	R/W	T_XUSB_DEV_XHCI_HSFSPI_COUNT13_U2_RESUME_K_DURATION: 1772h: U2_RESUME_K_DURATION_INIT (default)

22.16.15.48 T_XUSB_DEV_XHCI_HSFSPI_CTRL

Offset: 0x138 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	Reserved
30:26	1Fh	R/W	T_XUSB_DEV_XHCI_HSFSPI_CTRL_HS_IDLE_BIT_TIME: 1Fh: BIT_TIME_INIT
25:22	3Fh	R/W	T_XUSB_DEV_XHCI_HSFSPI_CTRL_FS_SE0_WIDTH: 3F: SE0_WIDTH_INIT
21	0	R/W	T_XUSB_DEV_XHCI_HSFSPI_CTRL_BITSTUFF_DISABLE: 0h: DISABLE_INIT
20	0	R/W	T_XUSB_DEV_XHCI_HSFSPI_CTRL_NRZI_DISABLE: 0h: DISABLE_INIT
19:10	3h	R/W	T_XUSB_DEV_XHCI_HSFSPI_CTRL_FS_INTERPKT_DELAY: 3h: DELAY_INIT
9:0	60h	R/W	T_XUSB_DEV_XHCI_HSFSPI_CTRL_HS_INTERPKT_DELAY: 60h: HS_INTERPKT_DELAY_INIT

22.16.15.49 T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_CTRL

Offset: 0x13C | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R	Reserved
1	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_CTRL_CYA_ADDR_MATCH: 0h CYA_ADDR_MATCH_INIT (default)
0	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_CTRL_PATTERN_SELECT: 0h PATTERN_SELECT_INIT (default)

22.16.15.50 T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_PATTERNx

PATTERN0 Offset: 0x140
PATTERN1 Offset: 0x144
PATTERN2 Offset: 0x148
PATTERN3 Offset: 0x14C
PATTERN4 Offset: 0x150
PATTERN5 Offset: 0x154
PATTERN6 Offset: 0x158
PATTERN7 Offset: 0x15C
PATTERN8 Offset: 0x160
PATTERN9 Offset: 0x164
PATTERN10 Offset: 0x168
PATTERN11 Offset: 0x16C
PATTERN12 Offset: 0x170
PATTERN13 Offset: 0x174
PATTERN14 Offset: 0x178
PATTERN15 Offset: 0x17C | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_PATTERNx_BYTE3: 0h BYTE3_INIT (default)
23:16	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_PATTERNx_BYTE2: 0h BYTE2_INIT (default)
15:8	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_PATTERNx_BYTE1: 0h BYTE1_INIT (default)
7:0	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_TESTMODE_PATTERNx_BYTE0: 0h BYTE0_INIT (default)

22.16.15.51 T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_CTRL

Offset: 0x180 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23:20	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_CTRL_END_MINOR: 0h END_MINOR_INIT (default)
19:16	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_CTRL_END_MAJOR: 0h END_MAJOR_INIT (default)
15:12	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_CTRL_START_MINOR: 0h START_MINOR_INIT (default)
11:8	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_CTRL_START_MAJOR: 0h START_MAJOR_INIT (default)
7:2	0	R	Reserved
1	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_CTRL_CLEAR: 0h CLEAR_NOT_PENDING (default) 1h CLEAR_PENDING 1h CLEAR_TRIGGER
0	0h	R/W	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_CTRL_RUN: 0h RUN_STOP (default) 1h RUN_RUN

22.16.15.52 T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS

Offset: 0x184 | Read/Write: R/W

Bits	Reset	R/W	Description
31:20	0	R	Reserved

Bits	Reset	R/W	Description
19	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_TESTMODE_HS_PKT:
18	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_TESTMODE_HS_NAK:
17	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_TESTMODE_HS_KSTATE:
16	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_TESTMODE_HS_JSTATE:
15	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_FS_RESUME:
14	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_FS_U3:
13	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_FS_U2:
12	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_FS_U0:
11	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_HS_RESUME:
10	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_HS_U3:
9	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_HS_U2:
8	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_ENABLED_HS_U0:
7	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_RESET_HS_PROLOG:
6	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_RESET_FS:
5	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_RESET_HS_RXCHIRPJ:
4	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_RESET_HS_RXCHIRPK:
3	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_RESET_HS_TXCHIRPK:
2	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_DISABLED_NONE:
1	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_DISCONNECTED_NONE:
0	Unknown	R	T_XUSB_DEV_XHCI_HSFSPI_PVTPORTDBG_STS_START_NONE:

22.16.15.53 T_XUSB_DEV_XHCI_SSPI_HOSTCFG_START

Offset: 0x600 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	1000h	R/W	T_XUSB_DEV_XHCI_SSPI_HOSTCFG_START_FIELD: 1000h FIELD_VALUE (default)

22.16.15.54 T_XUSB_DEV_XHCI_SSPI_HOSTCFG_END

Offset: 0x7FC | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0h	R/W	T_XUSB_DEV_XHCI_SSPI_HOSTCFG_END_FIELD: 0h FIELD_VALUE (default)

22.16.15.55 T_XUSB_DEV_XHCI_PERFMON_READ_CUMLATENCY_REG0

Offset: 0x800 | Read/Write: R/W

Bits	Reset	R/W	Description
15:0	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_CUMLATENCY_REG0_CYCLESHI: 0h CYCLESHI_INIT (default)
30:16	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_CUMLATENCY_REG0_PKTS: 0h PKTS_INIT (default)
31	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_CUMLATENCY_REG0_EN: 0h EN_INIT (default)

22.16.15.56 T_XUSB_DEV_XHCI_PERFMON_READ_CUMLATENCY_REG1

Offset: 0x804 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_CUMLATENCY_REG1_CYCLESLO: 0h CYCLESLO_INIT (default)

22.16.15.57 T_XUSB_DEV_XHCI_PERFMON_READ_LATENCY

Offset: 0x808 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_LATENCY_MINCYCLES: 0h MINCYCLES_INIT (default)
15:0	14h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_LATENCY_MAXCYCLES: 14h MAXCYCLES_INIT (default)

22.16.15.58 T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BOUNDARY_REG0

Offset: 0x80C | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BOUNDARY_REG0_B: 0h B_INIT (default)
15:0	14h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BOUNDARY_REG0_A: 14h A_INIT (default)

22.16.15.59 T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BOUNDARY_REG1

Offset: 0x810 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BOUNDARY_REG1_D: 0h D_INIT (default)
15:0	14h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BOUNDARY_REG1_C: 14h C_INIT (default)

22.16.15.60 T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG0

Offset: 0x814 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG0_1: 0h 1_INIT (default)
15:0	14h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG0_0: 14h 0_INIT (default)

22.16.15.61 T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG1

Offset: 0x818 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG1_3: 0h 3_INIT (default)
15:0	14h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG1_2: 14h 2_INIT (default)

22.16.15.62 T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG2

Offset: 0x81C | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	14h	R/W	T_XUSB_DEV_XHCI_PERFMON_READ_HISTOGRAM_BUCKET_REG2_4:

22.16.15.63 T_XUSB_DEV_XHCI_PERFMON_BI_CTRL

Offset: 0x820 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_BI_CTRL_EN: 0h EN_INIT (default)
30:0	0	R	Reserved

22.16.15.64 T_XUSB_DEV_XHCI_PERFMON_BI_EPTRB

Offset: 0x824 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	Reserved
24:16	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_BI_EPTRB_TRB: 0h TRB_INIT (default)
15:8	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_BI_EPTRB_EPSYSMEM: 0h EPSYSMEM_INIT (default)
7:0	0	R	Reserved

22.16.15.65 T_XUSB_DEV_XHCI_PERFMON_BI_DATA_OUT

Offset: 0x828 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_BI_DATA_OUT_REQCOUNT: 0h REQCOUNT_INIT (default)
23:0	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_BI_DATA_OUT_SIZE: 0h SIZE_INIT (default)

22.16.15.66 T_XUSB_DEV_XHCI_PERFMON_BI_DATA_IN

Offset: 0x82C | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_BI_DATA_IN_REQCOUNT: 0h REQCOUNT_INIT (default)
23:0	0h	R/W	T_XUSB_DEV_XHCI_PERFMON_BI_DATA_IN_SIZE: 0h SIZE_INIT (default)

22.16.15.67 T_XUSB_DEV_XHCI_BLCG

Offset: 0x840 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	Reserved
30	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_SS_PI_500M: 0h OVRD_SS_PI_500M_DISABLED (default) 1h OVRD_SS_PI_500M_ENABLED

Bits	Reset	R/W	Description
29	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_SS_PI: 0h OVRD_SS_PI_DISABLED (default) 1h OVRD_SS_PI_ENABLED
28	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_IOPLL_3_PWRDN: 0h OVRD_IOPLL_3_PWRDN_DISABLED (default) 1h OVRD_IOPLL_3_PWRDN_ENABLED
27	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_IOPLL_2_PWRDN: 0h OVRD_IOPLL_2_PWRDN_DISABLED (default) 1h OVRD_IOPLL_2_PWRDN_ENABLED
26	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_IOPLL_1_PWRDN: 0h OVRD_IOPLL_1_PWRDN_DISABLED (default) 1h OVRD_IOPLL_1_PWRDN_ENABLED
25	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_IOPLL_0_PWRDN: 0h OVRD_IOPLL_0_PWRDN_DISABLED (default) 1h OVRD_IOPLL_0_PWRDN_ENABLED
24	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_COREPLL_PWRDN: 0h OVRD_COREPLL_PWRDN_DISABLED (default) 1h OVRD_COREPLL_PWRDN_ENABLED
23	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_NVWRAP_48M: 0h OVRD_NVWRAP_48M_DISABLED (default) 1h OVRD_NVWRAP_48M_ENABLED
22	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_NVWRAP_480M: 0h OVRD_NVWRAP_480M_DISABLED (default) 1h OVRD_NVWRAP_480M_ENABLED
21	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_HSFS_PI: 0h OVRD_HSFS_PI_DISABLED (default) 1h OVRD_HSFS_PI_ENABLED
20	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_PICLK_BI: 0h OVRD_PICLK_BI_DISABLED (default) 1h OVRD_PICLK_BI_ENABLED
19	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_CORE_BI: 0h OVRD_CORE_BI_DISABLED (default) 1h OVRD_CORE_BI_ENABLED
18	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_FE: 0h OVRD_FE_DISABLED (default) 1h OVRD_FE_ENABLED
17	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_UFPCI: 0h OVRD_UFPCI_DISABLED (default) 1h OVRD_UFPCI_ENABLED
16	0h	R/W	T_XUSB_DEV_XHCI_BLCG_OVRD_DFPCI: 0h OVRD_DFPCI_DISABLED (default) 1h OVRD_DFPCI_ENABLED
15	0	R	Reserved
14	0h	R/W	T_XUSB_DEV_XHCI_BLCG_SS_PI_500M: 0h SS_PI_500M_DISABLED (default) 1h SS_PI_500M_ENABLED
13	0h	R/W	T_XUSB_DEV_XHCI_BLCG_SS_PI: 0h SS_PI_DISABLED (default) 1h SS_PI_ENABLED
12	0h	R/W	T_XUSB_DEV_XHCI_BLCG_IOPLL_3_PWRDN: 0h IOPLL_3_PWRDN_DISABLED (default) 1h IOPLL_3_PWRDN_ENABLED
11	0h	R/W	T_XUSB_DEV_XHCI_BLCG_IOPLL_2_PWRDN: 0h IOPLL_2_PWRDN_DISABLED (default) 1h IOPLL_2_PWRDN_ENABLED
10	0h	R/W	T_XUSB_DEV_XHCI_BLCG_IOPLL_1_PWRDN: 0h IOPLL_1_PWRDN_DISABLED (default) 1h IOPLL_1_PWRDN_ENABLED
9	0h	R/W	T_XUSB_DEV_XHCI_BLCG_IOPLL_0_PWRDN: 0h IOPLL_0_PWRDN_DISABLED (default) 1h IOPLL_0_PWRDN_ENABLED

Bits	Reset	R/W	Description
8	0h	R/W	T_XUSB_DEV_XHCI_BLCG_COREPLL_PWRDN: 0h COREPLL_PWRDN_DISABLED (default) 1h COREPLL_PWRDN_ENABLED
7	0h	R/W	T_XUSB_DEV_XHCI_BLCG_NVWRAP_48M: 0h NVWRAP_48M_DISABLED (default) 1h NVWRAP_48M_ENABLED
6	0h	R/W	T_XUSB_DEV_XHCI_BLCG_NVWRAP_480M: 0h NVWRAP_480M_DISABLED (default) 1h NVWRAP_480M_ENABLED
5	0h	R/W	T_XUSB_DEV_XHCI_BLCG_HSFS_PI: 0h HSFS_PI_DISABLED (default) 1h HSFS_PI_ENABLED
4	0h	R/W	T_XUSB_DEV_XHCI_BLCG_PICLK_BI: 0h PICLK_BI_DISABLED (default) 1h PICLK_BI_ENABLED
3	0h	R/W	T_XUSB_DEV_XHCI_BLCG_CORE_BI: 0h CORE_BI_DISABLED (default) 1h CORE_BI_ENABLED
2	0h	R/W	T_XUSB_DEV_XHCI_BLCG_FE: 0h FE_DISABLED (default) 1h FE_ENABLED
1	0h	R/W	T_XUSB_DEV_XHCI_BLCG_UFPCI: 0h UFPCI_DISABLED (default) 1h UFPCI_ENABLED
0	0h	R/W	T_XUSB_DEV_XHCI_BLCG_DFPCI: 0h DFPCI_DISABLED (default) 1h DFPCI_ENABLED

22.16.15.68 _XUSB_DEV_XHCI_BLCG_STS

Offset: 0x844 | Read/Write: R/W

Bits	Reset	R/W	Description
31:15	0	R	Reserved
14	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_SS_PI_500M: 0h SS_PI_500M_DISABLED (default) 1h SS_PI_500M_ENABLED
13	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_SS_PI: 0h SS_PI_DISABLED (default) 1h SS_PI_ENABLED
12	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_IOPLL_3_PWRDN: 0h IOPLL_3_PWRDN_DISABLED (default) 1h IOPLL_3_PWRDN_ENABLED
11	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_IOPLL_2_PWRDN: 0h IOPLL_2_PWRDN_DISABLED (default) 1h IOPLL_2_PWRDN_ENABLED
10	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_IOPLL_1_PWRDN: 0h IOPLL_1_PWRDN_DISABLED (default) 1h IOPLL_1_PWRDN_ENABLED
9	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_IOPLL_0_PWRDN: 0h IOPLL_0_PWRDN_DISABLED (default) 1h IOPLL_0_PWRDN_ENABLED
8	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_COREPLL_PWRDN: 0h COREPLL_PWRDN_DISABLED (default) 1h COREPLL_PWRDN_ENABLED
7	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_NVWRAP_48M: 0h NVWRAP_48M_DISABLED (default) 1h NVWRAP_48M_ENABLED
6	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_NVWRAP_480M: 0h NVWRAP_480M_DISABLED (default) 1h NVWRAP_480M_ENABLED

Bits	Reset	R/W	Description
5	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_HSFS_PI: 0h HSFS_PI_DISABLED (default) 1h HSFS_PI_ENABLED
4	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_PICLK_BI: 0h PICLK_BI_DISABLED (default) 1h PICLK_BI_ENABLED
3	0h	R	T_XUSB_DEV_XHCI_BLCG_STS_CORE_BI: 0h CORE_BI_DISABLED (default) 1h CORE_BI_ENABLED
2:0	0	R	Reserved

22.16.15.69 T_XUSB_DEV_XHCI_BLCG_INTR

Offset: 0x848 | Read/Write: R/W

Bits	Reset	R/W	Description
31:29	0	R	Reserved
28	0h	R	T_XUSB_DEV_XHCI_BLCG_INTR_STS_IOPLL_3_PWRDN: 0h STS_IOPLL_3_PWRDN_DISABLED (default) 1h STS_IOPLL_3_PWRDN_ENABLED
27	0h	R	T_XUSB_DEV_XHCI_BLCG_INTR_STS_IOPLL_2_PWRDN: 0h STS_IOPLL_2_PWRDN_DISABLED (default) 1h STS_IOPLL_2_PWRDN_ENABLED
26	0h	R	T_XUSB_DEV_XHCI_BLCG_INTR_STS_IOPLL_1_PWRDN: 0h STS_IOPLL_1_PWRDN_DISABLED (default) 1h STS_IOPLL_1_PWRDN_ENABLED
25	0h	R	T_XUSB_DEV_XHCI_BLCG_INTR_STS_IOPLL_0_PWRDN: 0h STS_IOPLL_0_PWRDN_DISABLED (default) 1h STS_IOPLL_0_PWRDN_ENABLED
24	0h	R	T_XUSB_DEV_XHCI_BLCG_INTR_STS_COREPLL_PWRDN: 0h STS_COREPLL_PWRDN_DISABLED (default) 1h STS_COREPLL_PWRDN_ENABLED
23:13	0	R	Reserved
12	0h	R/W	T_XUSB_DEV_XHCI_BLCG_INTR_IOPLL_3_PWRDN: 0h IOPLL_3_PWRDN_DISABLED (default) 1h IOPLL_3_PWRDN_ENABLED
11	0h	R/W	T_XUSB_DEV_XHCI_BLCG_INTR_IOPLL_2_PWRDN: 0h IOPLL_2_PWRDN_DISABLED (default) 1h IOPLL_2_PWRDN_ENABLED
10	0h	R/W	T_XUSB_DEV_XHCI_BLCG_INTR_IOPLL_1_PWRDN: 0h IOPLL_1_PWRDN_DISABLED (default) 1h IOPLL_1_PWRDN_ENABLED
9	0h	R/W	T_XUSB_DEV_XHCI_BLCG_INTR_IOPLL_0_PWRDN: 0h IOPLL_0_PWRDN_DISABLED (default) 1h IOPLL_0_PWRDN_ENABLED
8	0h	R/W	T_XUSB_DEV_XHCI_BLCG_INTR_COREPLL_PWRDN: 0h COREPLL_PWRDN_DISABLED (default) 1h COREPLL_PWRDN_ENABLED
7:1	0	R	Reserved
0	0h	R/W	T_XUSB_DEV_XHCI_BLCG_INTR_TGT: 0h TGT_SMI (default) 1h TGT_PME

22.16.15.70 T_XUSB_DEV_XHCI_CFG_DEVBI

Offset: 0x850 | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved

Bits	Reset	R/W	Description
29	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_ISOCH_SKIP_SIA: 1h ISOCH_SKIP_SIA_INIT (default)
28:24	10h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_DMA_RD_MAX_ALOM: 10h DMA_RD_MAX_ALOM_INIT (default)
23:16	26h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_CNT_250NS: 26h CNT_250NS_INIT (default)
15	0	R	Reserved
14	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_TRBFETCH_RDPASSPW: 0h TRBFETCH_RDPASSPW_INIT (default)
13	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_ASYNC_EP_IDLE: 1h ASYNC_EP_IDLE_INIT (default)
12	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_LOCAL_ROTATE: 1h LOCAL_ROTATE_INIT (default) 1h LOCAL_ROTATE_EN 0h LOCAL_ROTATE_DIS
11	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_TRBFETCH_RINGEND_CHK: 1h TRBFETCH_RINGEND_CHK_EN (default)
10:9	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_TRBFETCH_NUMSKIP: 1h TRBFETCH_NUMSKIP_INIT (default)
8	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_TRBFETCH_IDT_IN: 0h TRBFETCH_IDT_IN_INIT (default)
7	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_DMA_WR_UPSTREAM_RO: 1h DMA_WR_UPSTREAM_RO_INIT (default)
6	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_DMA_RD_UPSTREAM_RO: 1h DMA_RD_UPSTREAM_RO_INIT (default)
5	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_DMA_RD_UPSTREAM_RDPASSPW: 0h DMA_RD_UPSTREAM_RDPASSPW_INIT (default)
4:0	3h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_DMA_WR_MAX_ALOM: 3h DMA_WR_MAX_ALOM_INIT (default)

22.16.15.71 T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM

Offset: 0x854 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23:16	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_DMA_RD_LIMIT: 0h DMA_RD_LIMIT_INIT (default)
15:12	0	R	Reserved
11	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_EPLOGIC_RDPASSPW: 0h EPLOGIC_RDPASSPW_INIT (default)
10	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_EPLOGIC_RO: 0h EPLOGIC_RO_INIT (default)
9	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_EPLOGIC_NS: 0h EPLOGIC_NS_INIT (default)
8	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_EPLOGIC_TC: 0h EPLOGIC_TC_INIT (default)
7:3	0	R	Reserved
2	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_EVENTQ_RO: 0h EVENTQ_RO_INIT (default)
1	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_EVENTQ_NS: 0h EVENTQ_NS_INIT (default)
0	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEVBI_UPSTREAM_EVENTQ_TC: 0h EVENTQ_TC_INIT (default)

22.16.15.72 T_XUSB_DEV_XHCI_CFG_DEV_SSPI_XFER

Offset: 0x858 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	F00h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_SSPI_XFER_ACKTIMEOUT: F00h ACKTIMEOUT_INIT (default)

22.16.15.73 T_XUSB_DEV_XHCI_CFG_DEV_FE

Note: *Do not change the values in bits 31:2*

Offset: 0x85C | Read/Write: R/W

Bits	Reset	R/W	Description
31:30	0	R	Reserved
29	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_INFINITE_SS_RETRY: 1h INFINITE_SS_RETRY_EN 0h INFINITE_SS_RETRY_DIS (default)
28	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_EN_PRIME_EVENT: 1h EN_PRIME_EVENT_INIT (default)
27	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_FEATURE_LPM: 1h FEATURE_LPM_EN (default) 0h FEATURE_LPM_DIS
26	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_EN_STALL_EVENT: 0h EN_STALL_EVENT_INIT (default)
25	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_PORTDISCON_RST_HW: 1h PORTDISCON_RST_HW_INIT (default)
24	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_CTX_RESTORE: 0h CTX_RESTORE_INIT (default)
23:4	4B0h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_MFCOUNT_MIN: 4B0h MFCOUNT_MIN_INIT (default)
3	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_PORTRST_HW: 1h PORTRST_HW_INIT (default)
2	1h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_SEQNUM_INIT: 1h SEQNUM_INIT_INIT (default)
1:0	0h	R/W	T_XUSB_DEV_XHCI_CFG_DEV_FE_PORTREGSEL: Can be used to override the register selection between USB2 and USB3 versions of the PORTSC and PORTPM registers. By default, the register accesses are routed based on the current speed of operation. This field allows software to access USB2 or USB3 port registers specifically. 0: Default. Access gets routed to correct register based on current link speed. 1: SS. Overrides access to SS PortSC Register. 2: HS. Overrides access to HSFS PortSC Register. 3: Reserved 0h PORTREGSEL_INIT (default) 1h PORTREGSEL_SS 2h PORTREGSEL_HSFS

22.16.15.74 T_XUSB_DEV_XHCI_CFG_IDLE

Offset: 0x860 | Read/Write: R/W

Bits	Reset	R/W	Description
31:5	0	R	Reserved
4	None	R	T_XUSB_DEV_XHCI_CFG_IDLE_DEV_SS_PI:
3	None	R	T_XUSB_DEV_XHCI_CFG_IDLE_DEV_FS_NVWRAP:
2	None	R	T_XUSB_DEV_XHCI_CFG_IDLE_DEV_HS_NVWRAP:
1	None	R	T_XUSB_DEV_XHCI_CFG_IDLE_DEV_HSFS_PI:

Bits	Reset	R/W	Description
0	None	R	T_XUSB_DEV_XHCI_CFG_IDLE_DEV_BI:

22.16.15.75 T_XUSB_DEV_XHCI_CFG_SSXFER

Offset: 0x864 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	3938700h	R/W	T_XUSB_DEV_XHCI_CFG_SSXFER_ERDYTIMER: 3938700h ERDYTIMER_INIT (default)

22.16.15.76 T_XUSB_DEV_XHCI_CFG_SSXFER1

Offset: 0x868 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	2h	R/W	T_XUSB_DEV_XHCI_CFG_SSXFER1_PINGTIMER: 2h PINGTIMER_INIT (default)

22.16.15.77 T_XUSB_DEV_XHCI_CFG_SPARE0

Offset: 0x86C | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0h	R/W	T_XUSB_DEV_XHCI_CFG_SPARE0_REG: 0h REG_INIT (default)

22.16.15.78 T_XUSB_DEV_XHCI_CFG_SPARE1

Offset: 0x870 | Read/Write: R/W

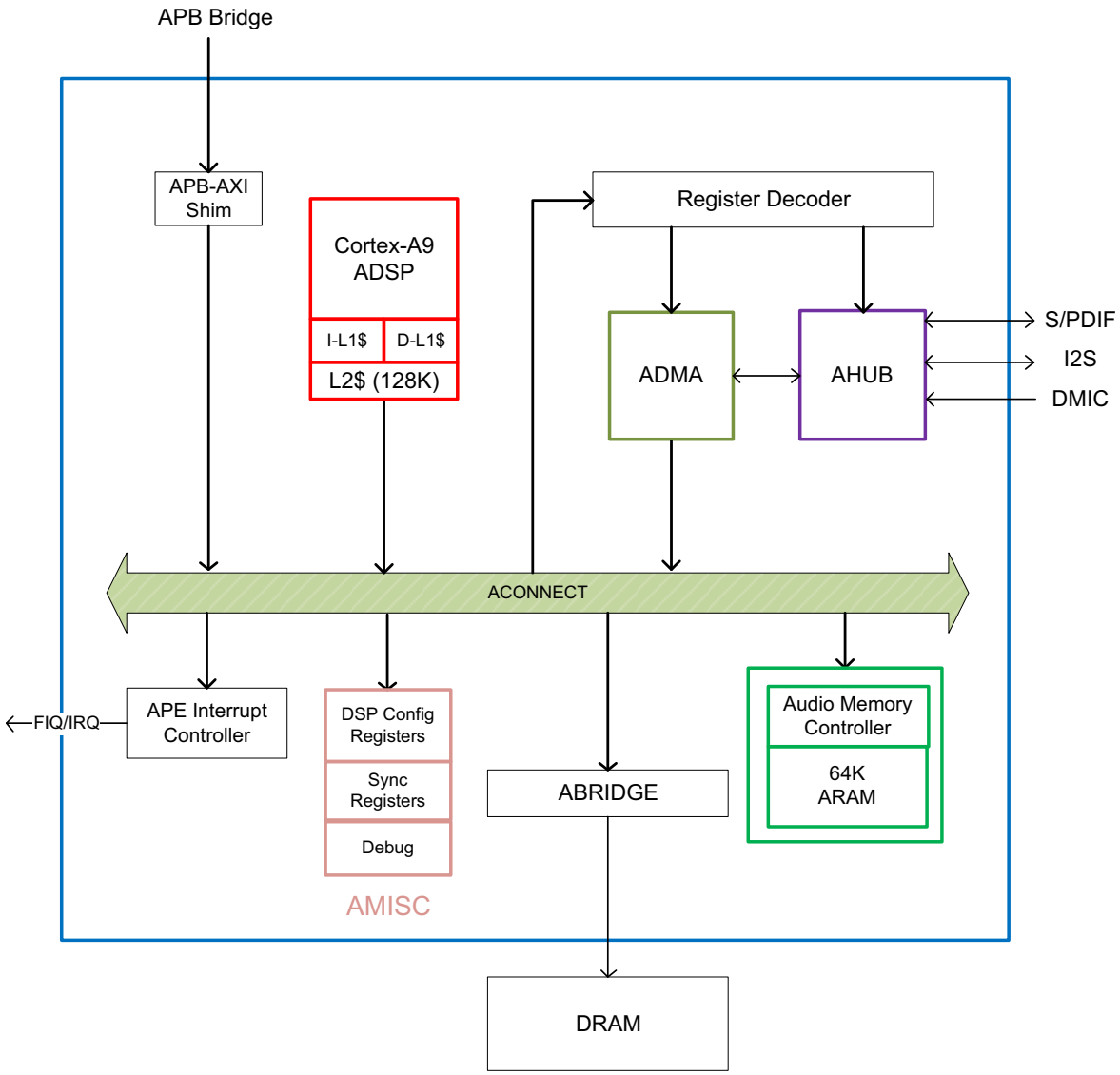
Bits	Reset	R/W	Description
31:0	7FFFFFFFh	R/W	T_XUSB_DEV_XHCI_CFG_SPARE1_REG: 7FFFFFFFh REG_INIT (default)

CHAPTER 23: AUDIO PROCESSING ENGINE

The audio processing engine (APE) in Tegra® X1 mobile processors takes care of all the audio needs of Tegra chips with minimal supervision from the CPU. The APE contains an audio DSP (ADSP), an internal RAM, an “Audio Hub” or AHUB containing many hardware accelerators, and a DMA engine as shown below. The AHUB has external I2S, S/PDIF, and DMIC interfaces. The AHUB supports multiple interfaces to the audio devices in the system cellular baseband; different audio codecs; Bluetooth modules; A/V receivers; etc. The AHUB can support the different interface, protocol, and signal quality requirements of these audio devices.

Note: *For the APE address map, see Section 2.2: APE Address Map.*

Figure 65: Audio Processing Engine Functional Block Diagram



The audio processing engine (APE) provides these interfaces:

- APB Interface
- Audio Interfaces:
  - I2S x 5
  - SPDIF x1
  - DMIC x 3
- Interrupts
  - IRQCPU, FIQCPU – Interrupts to CPU
- Debug Port
  - CoreSight™ interface to ADSP
  - UART (no DMA engine)
- Clocks
  - MC_CLK
  - PCLK (apb_clock)
  - APE_CLK
  - ADSP_CLK
- Resets
  - APE_reset
  - ADSP_reset
  - PCLK
- Memory Interface
- External Signaling
  - 56-bit gray scale TSC signals

23.1 APE Overview

APE 1.0 consists of the following modules:

- ADSP includes the ARM based CPU and its cache
- AHUB contains the audio hardware accelerators. It also interfaces with audio I/O
- ADMA is responsible for moving data between AHUB and DRAM/ARAM
- REGDEC serves as the register interface for ADMA and AHUB
- ACONNECT is an AXI based backbone which directs requests from different masters to targeted slaves
- AMC is the local memory controller for local memory residing in this module
- ABRIDGE is the memory interface to DRAM
- AGIC is the interrupt controller
- AMISC contains the configuration registers and synchronizations registers
- AAS converts APB transactions from CPU to AXI transactions

23.1.1 Audio Digital Signal Processor (ADSP)

The ADSP contains the following modules:

- Cortex®-A9 ADSP with 32K-I/32K-D L1 cache and NEON extension
- PL310 L2 Cache controller
- 128KB L2 cache

Coherency

Coherent Request: A read or write request from ADSP to main memory. Coherent indicates that the CPU's caches will be checked for modified data.

Non-Coherent Request: A read or write request from ADSP to main memory. Coherent indicates that the CPU's caches will NOT be checked for modified data.

CPU producer – ADSP consumer

- Example use case is normal audio decode by ADSP:
  - CPU reads file data and writes into <file_buffer> in main memory.
  - ADSP reads <file_buffer> from main memory and processes data
- Assumptions:
  - ADSP does not need to modify data in <file_buffer>
- Recommended Settings:
  - CPU: Cacheable Write-Back

    Don't have to go to DRAM. Reduced latency/power.
  - ADSP: Cacheable Read-Allocate
    - CPU reuses buffer for new data and pings ADSP to invalidate the data in the cache.
    - The data in ADSP cache has to be flushed/invalidated when the CPU reuses the buffer in main memory. Otherwise, ADSP would access stale data.
  - DMA
    - DMA copy from memory into ARAM.
    - ADSP reads ARAM cacheable.
    - Increased amount of requests by through ACONNECT (meaning there is more memory traffic through ACONNECT if DMA transfers data from memory to ARAM first and ADSP reads from ARAM vs. ADSP reading from memory directly).

      DMA RD

      DMA WR

      ADSP RD

ADSP producer – CPU consumer

- Example use case: Audio Recording
  - ADSP processes and encodes data, writes to <file_buffer>
  - CPU reads <file_buffer> from main memory
- Assumptions:
  - CPU does not need to modify data in <file_buffer>
- Recommended Settings:
  - ADSP: Write-Through or Non-Cacheable
    - Both of these instances will result in sub-cacheline writes. This hurts bandwidth.

- DMA
  - ADSP writes to ARAM.
  - DMA copies from ARAM to memory
  - Increased traffic through ACONNECT

ADSP – CPU sharing

Both CPU/BPMP-Lite can write/modify data

Memory Aliasing due to TrustZone bit

PL310 L2 cache controller supports TrustZone accesses. It treats NS bit as an address tag bit. A secure and a non-secure to the same physical memory location will result in 2 separate copies in the cache. If data is modified inside the cache, data corruption could result. Software needs to be aware of this data coherency issue. Note that ABRIDGE and AMC are non-secure slaves and simply ignore the NS bit.

Exclusive Cache Configuration

Cortex-A9 L1 and PL310 L2 are configured as inclusive cache by default. Exclusive cache is supported by modifying the exclusive cache bit in ACTLR of A9 and bit 12 in Auxiliary Control Register of PL310. In exclusive, a cache line is either valid in L1 or L2 but not both. See the CoreLink™ Level 2 Cache Controller L2C-310, Revision r3p3, Section 2.3.4 for cache behavior.

It is recommended that ADSP L1 and L2 are configured as exclusive cache to maximize the amount of data cached.

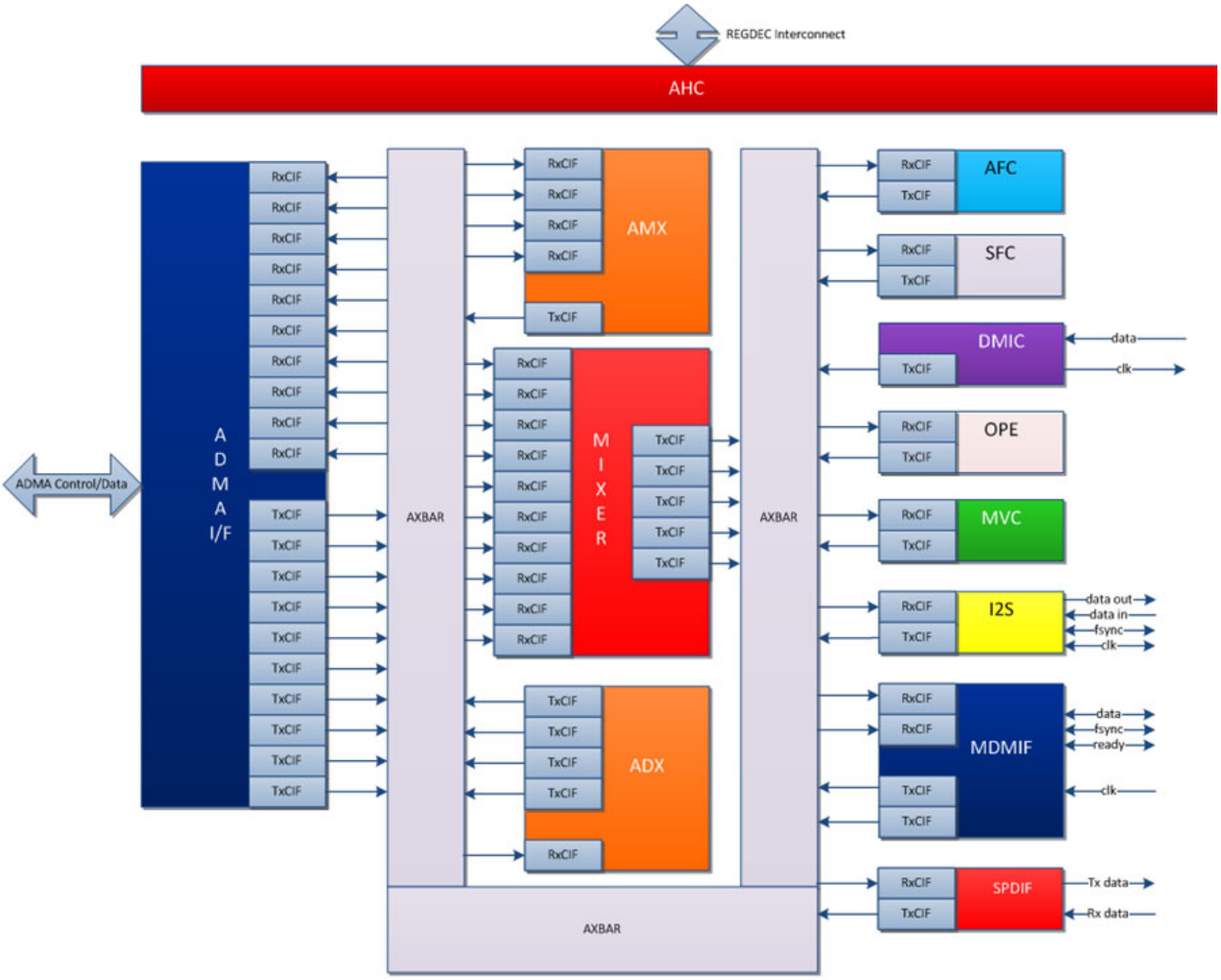
Cache Lockdown

L2 cache supports “lockdown by way”. PL310 provides bit mask that specifies which way can be allocated for cache line fill. There is a separate bitmask for data and instruction cache.

23.1.2 AHUB

Audio Hub (AHUB) is a collection of hardware accelerators with a means for audio data to be routed through these accelerators. The purpose of AHUB is to offload repetitive tasks that do not need constant decision-making from CPU/Audio DSP (ADSP).

Figure 66: AHUB Block Diagram



The hardware accelerators contained in the AHUB are:

- Inter-IC Sound Controller (I2S)
- Digital MIC Controller (DMIC)
- Sony/Philips Digital Interface Controller (SPDIF)
- Mixer
- Audio Multiplexer (AMX)
- Audio De-Multiplexer (ADX)
- Sampling Frequency Converter (SFC)
- Audio Flow Controller (AFC)
- Output Processing Engine (OPE)
- Master Volume Control (MVC)
- Audio Direct Memory Access Interface (ADMA I/F)

A proprietary interface called Audio Client Interface (ACIF) is used to route audio samples through these accelerators and hence forms the fabric of AHUB. A switch called Audio Crossbar (AXBAR) is used to configure/modify the audio routing path

between these accelerators. The accelerator modules and the audio routing are both configured via an AHUB Configuration (AHC) unit.

23.1.2.1 Audio Client Interface (ACIF)

Audio streams are routed through the AHUB by interconnecting various modules using the Audio Client Interface (ACIF).

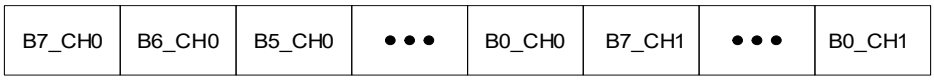
ACIF Protocol

Before a session starts, both TX and RX clients should know the frame length by their register configurations. The frame length is determined by the number of bits in each sample and the number of channels that the audio data carries. In the case of stereo audio data with 16 bit, a frame is 32 bits long.

Whenever TX has data available to transmit to the RX, it raises the fsync signal. Simultaneously the first bit of the data is put out on the data bus. The TX should hold this *fsync* active for at least one clock but until the RX client is ready. Similarly, it should also hold the data bus to the first bit of the frame until the RX client is ready. When the RX asserts ready, the TX should lower the fsync at the end of that clock cycle. From the next clock cycle, TX should transmit the rest of the frame (from second bit onwards), 1 bit for each bit clock.

The figure below shows the transmission ordering of 2-channel 8-bit audio data. Note that the bit ordering is each channel is big-endian.

Figure 67: Bit Ordering



Frame Length

The frame length is simply a product of the bit width and the number of channels in an audio stream. The register fields of the TX and RX clients should be programmed accordingly before the transmission starts to avoid incorrect frame parsing.

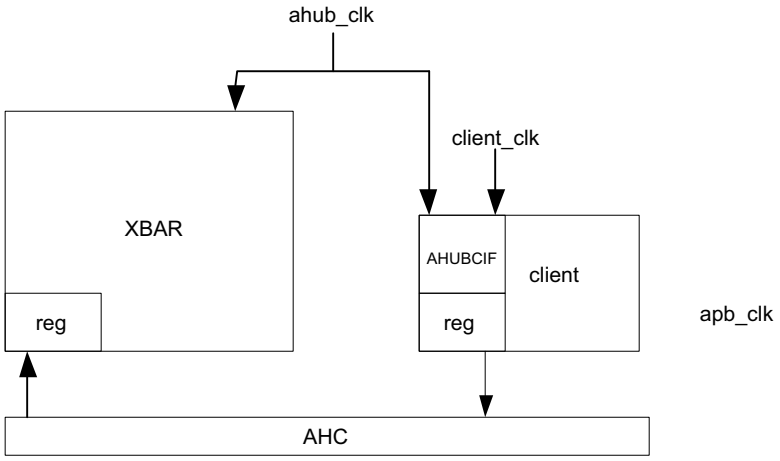
Clocking

Since all sessions are AHUB run under an AHUB clock, the clock should be fast enough to transmit audio stream data in all sessions. If a session is used for sending 8 kHz, stereo 16-bit audio stream, the AHUB clock should be at least 256 kHz. While a faster clock is better for data transmission, it is not necessarily good for power consumption. For the typical audio application like MP3 music playing, using the clock directly from the crystal without going through PLLs is recommended. Depending on the applications, users can program the clock registers to satisfy the requirement of clock frequency. The following table shows the various clock frequencies for AHUB.

Clock Frequency	Description
64 kHz	Slowest clock to carry 8-bit mono voice data in 8kHz sampling
1.4112 MHz	Clock to carry 16-bit stereo audio data in 44.1 kHz sampling
1.536 MHz	Clock to carry 16-bit stereo audio data in 48 kHz sampling
4.608 MHz	Clock to carry 24-bit stereo audio data in 96 kHz sampling or 16-bit 6 channel audio data in 48 kHz sampling
12 MHz	Minimum crystal frequency
18.432 MHz	Clock to carry 24-bit 8 channel audio data in 96 kHz sampling
36.864 MHz	Clock to carry 24-bit 8 channel audio data in 192 kHz sampling (HDMI maximum)

While AHUB runs under one clock, *ahub_clk,* its clients have their own clocks. AHUBCIF is the clock boundary in the clients. The following figure shows the clocking scheme around AHUB.

Figure 68: Clock Domains



Packing/Unpacking Data

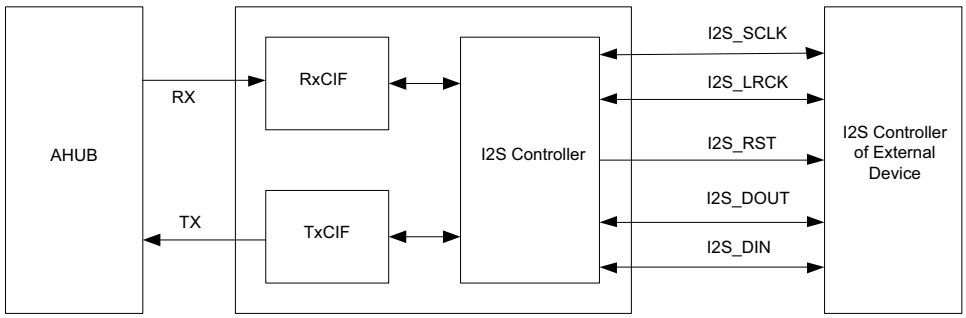
This feature is only available for ADMA Interface. The TxCIFs have the capability to unpack 32-bit packet (coming from ADMA) into smaller data of 8 or 16 bits before transmitting to the AXBAR. The RxCIFs have the reverse capability of packing 8- or 16-bit words into a 32-bit packet.

The caveat is that, in the packing can only be done if the incoming bits are 8-bit or 16-bit. It is wrong programming to try to pack data that is 24 bits. Similarly one cannot unpack 32-bit data into 24-bit data. So the AXBAR_BITS in both cases can be only 8 or 16.

23.1.2.2 I^2S Controller

The Inter-IC Sound (I^2S) controller implements full-duplex and bidirectional and single direction point-to-point serial interfaces. It can interface with I^2S-compatible products, such as compact disc players, digital audio tape devices, digital sound processors, modems, Bluetooth chips, etc.

Figure 69: I2S Interaction with External Device



The I^2S controller can operate both as master and slave. It supports the following data transfer modes:

- I^2S mode
- Left Justified Mode (LJM)
- Right Justified Mode (RJM)
- DSP mode, as defined in the Philips inter-IC-sound (I^2S) bus specification
- PCM mode with short (one-bit-clock wide) and long-fsync (two bit-clocks wide
- Network (Telephony) mode with independent slot selection for both Tx and Rx

- TDM mode with flexibility in number of slots with up to 16 slots.
- Capability to drive-out a High-Z outside the prescribed slot for transmission

The I^2S controller can transmit and receive word lengths of 8, 16, 24, and 32.

It supports u-Law and A-Law compression/decompression.

The I^2S controller can control the flow of traffic from another I^2S controller operating on an independent bit clock (with a ppm difference compared with its own bit clock).

Transmission/Reception Data Formats

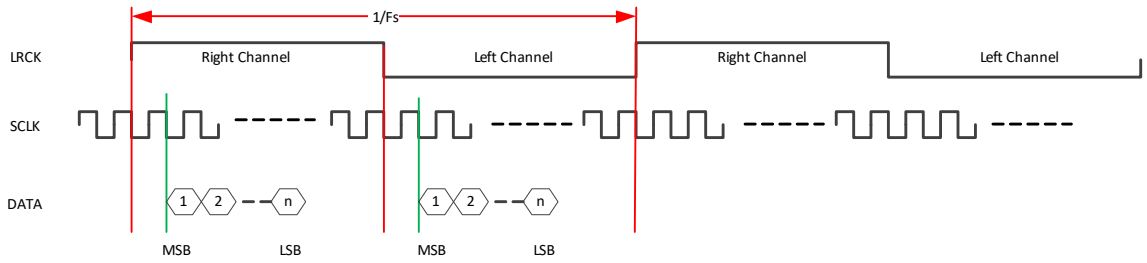
LRCK Modes

This subsection illustrates three LRCK modes: Basic I2S mode, Right Justified mode, and Left Justified mode.

Basic I2S Mode

In Basic I2S mode, data starts one sclk after the LRCK edge (offset = 1).

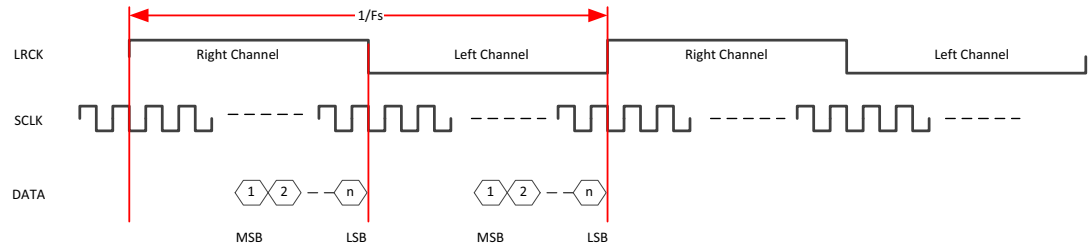
Figure 70: Basic I2S Mode



Right Justified (RJ) Mode

In RJ mode, data starts (r/2 – n) SCLKs after the LRCK edge (offset = r/2 – n, where r = number of SCLKs per LRCK, n = number of bits/sample).

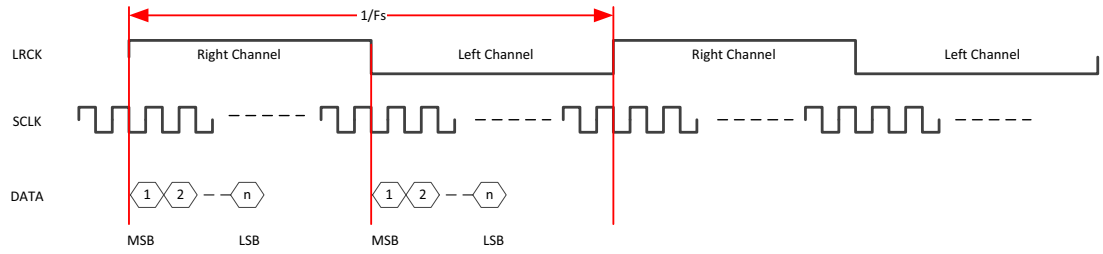
Figure 71: RJ Mode



Left Justified (LJ) Mode

In LJ mode, data starts 0 SCLKs after the LRCK edge (offset = 0).

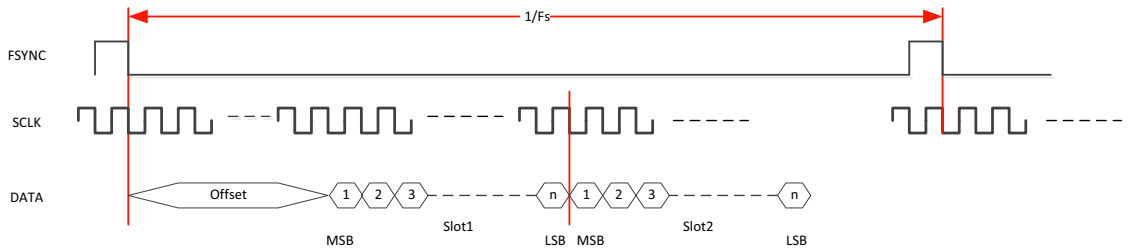
Figure 72: LJ Mode



FSYNC Modes

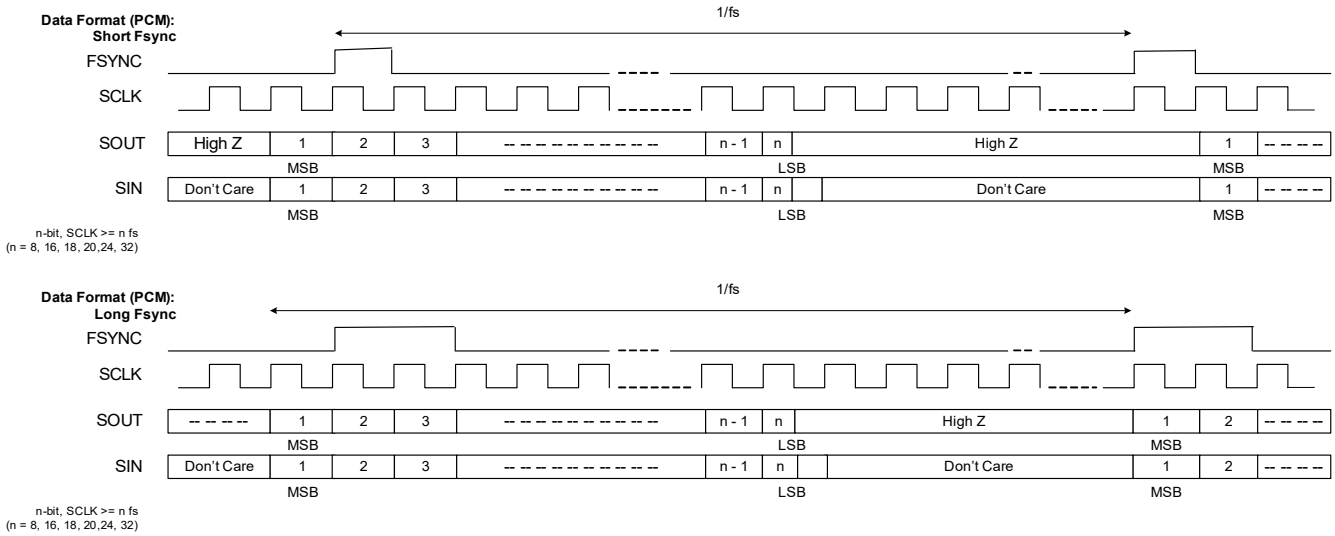
The width of the FSYNC, the offset value, number of slots and number of SCLKs per 1/Fs are all configurable. The slots are always contiguous.

Figure 73: FSync Mode

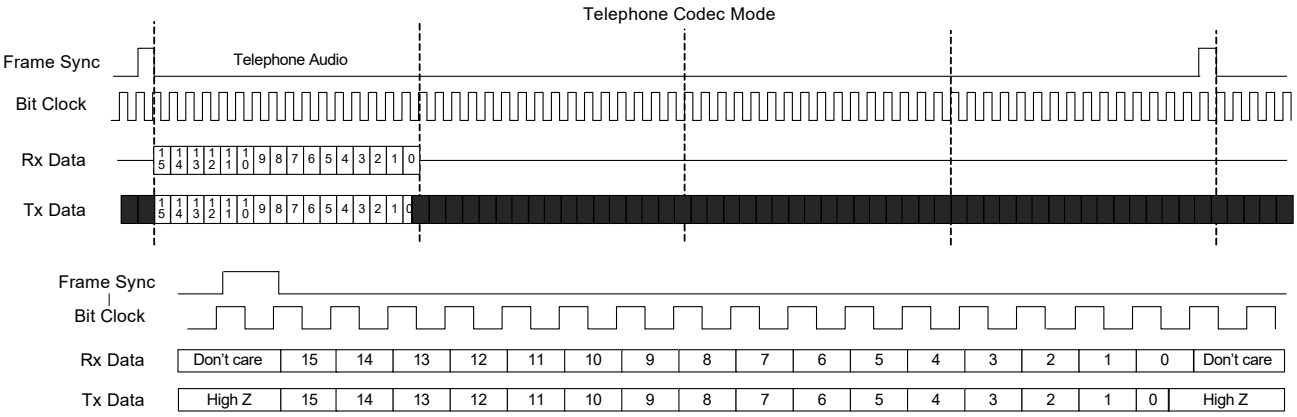


PCM Mode

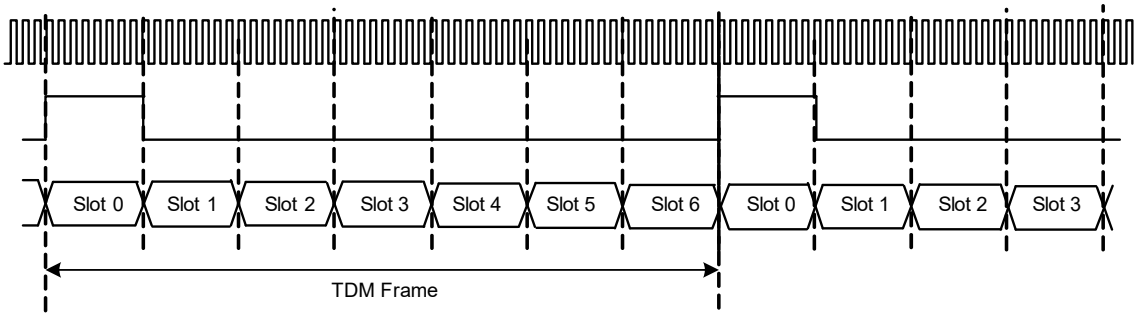
Figure 74: PCM Mode



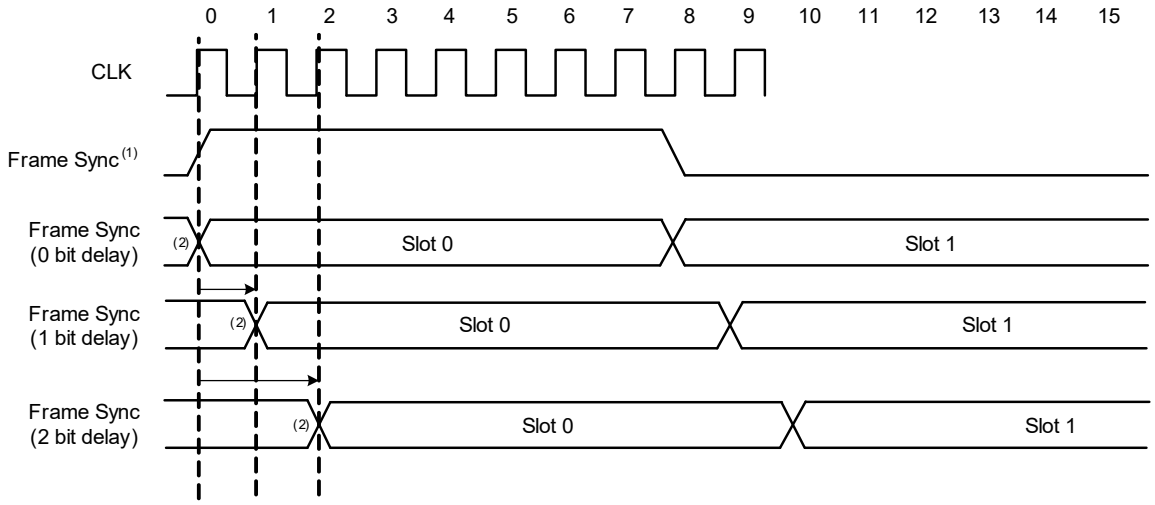
Data Formats in NW Mode



Data Format in TDM Mode



1. FS duration of a slot is shown. FS duration of a single bit is also supported.



1. FS duration of a slot is shown. FS duration of a single bit is also supported.
2. Last bit of the last slot of the previous frame. No gap is allowed between this bit and the first bit of slot 0.

23.1.2.3 Digital MIC Controller (DMIC)

The DMIC Controller is used to interface with PDM base input devices. The DMIC controller implements a converter to convert PDM (Pulse density modulation) signals to PCM (Pulse code modulation) signals.

DMIC Features

- Sample rate support: 8 kHz- 48 kHz
- Input PCM bit width: 16-24 bits
- Oversampling Ratio: 64, 128, 256

23.1.2.4 Serial Peripheral Device Interface (S/PDIF)

This interface is primarily intended to carry audio data coded other than as linear PCM coded audio samples. Provisions are also made to allow the interface to carry data related to computer software or signals coded using non-linear PCM. The format specification for these applications is not part of this standard.

S/PDIF Features

- Fully implements the IEC-958 standard interface
- Provides mono and stereo support for sampling frequencies 32 kHz, 44.1 kHz, and 48 kHz
- Supports the following data formats.
  - 16-bit

- 20-bit
- 24-bit
- Raw

- Supports “autolock” mode to automatically detect the “spdifin” sample rate and lock onto the data stream.
- Supports “override” mode to provide a manual control to sample the “spdifin” data stream.
- Provides a loopback mode to route the “spdifout” back to “spdifin” for self-testing.

23.1.2.5 Mixer

The Mixer is a key module in the audio architecture. Whether it is a smart phone/tablet environment or an automotive solution, mixing of different audio signals is an often used feature. Some examples are mixing system tones with voice calls, mixing two audio playbacks to create a transition effect, mixing Sat Nav announcements with audio playback etc.

Mixer Features

- Supports mixing up to 10 input streams of 7.1 channel audio each
- Supports 5 outputs each of which can be a mix of any combination of 10 input streams
- Time ramp-up/ramp-down volume control provided for each stream
- Fixed gain for each stream is also available
- A 32-bit sample counter is provided for each input stream to count the number of samples consumed
- A peak meter for each input stream is available. It can give peak values for non-overlapping frames of samples or can do continuous reset-on-read peak metering

23.1.2.6 Audio Multiplexer Block (AMX)

The Audio Multiplexer Block (AMX) can multiplex up to 4 input streams of up to 16 channels each and generate an output stream of up to 16 channels

A “byte RAM” helps to form an output frame by any combination of bytes from the 4 input frames

Two modes for data synchronization between input frames are available:

- Wait For All mode: At the beginning, wait for all enabled input streams to have data before forming the very first frame.
- Wait for Any mode: Start whenever data is available in any one of the enabled input streams.

In either mode, once the first output frame is sent out, AMX always waits for all active streams to have data available before forming and sending subsequent frames.

23.1.2.7 Demultiplexer Block (ADX)

The demultiplexer block (ADX) takes an input stream with up to 16 channels and demultiplexes it into four output streams of up to 16 channels each.

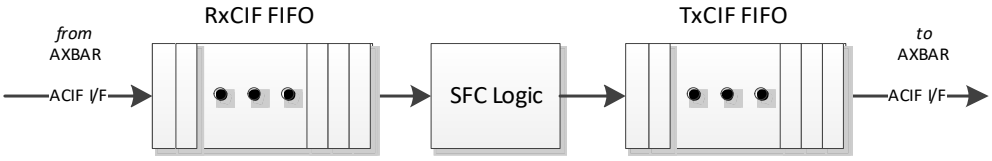
A “byte RAM” helps to form output frames by any combination of bytes from the input frame. Its design is identical to that of the byte RAM in the AMX except that the data flow direction is reversed.

23.1.2.8 Sampling Frequency Converter (SFC)

The sampling frequency converter (SFC) converts the sampling frequency of the input signal from one frequency to another.

The SFC is an AXBAR client and hence interacts with other AHUB modules using the Audio Client Interface (ACIF) protocol. Hence the SFC has a Receive CIF (RxCIF) and Transmit CIF (TxCIF), each with its own FIFO as shown below.

Figure 75: SFC's Receive and Transmit CIFs



SFC Features

- Supports sampling frequency conversion of streams of up to 2 channels (stereo)
- Has a very low latency with the maximum latency of < 125 µs
- Supports the following frequency conversions

FsIn\FsOut	8	11.025	16	22.05	24	32	44.1	48	88.2	96	176.4	192
8	Bypass	x	x	x	x	x	x	x	x	x		
11.025	x	Bypass	x	x	x	x	x	x	x	x		
16	x	x	Bypass	x	x	x	x	x	x	x	x	x
22.05	x	x	x	Bypass	x	x	x	x	x	x	x	x
24	x	x	x	x	Bypass	x	x	x	x	x	x	x
32	x	x	x	x	x	Bypass	x	x	x	x	x	x
44.1	x	x	x	x	x	x	Bypass	x	x	x	x	x
48	x	x	x	x	x	x	x	Bypass	x	x	x	x
88.2	x	x	x	x	x	x	x	x	Bypass	x	x	x
96	x	x	x	x	x	x	x	x	x	Bypass	x	x
176.4			x	x	x	x	x	x	x	x	Bypass	x
192			x	x	x	x	x	x	x	x	x	Bypass

23.1.2.9 Audio Flow Controller (AFC)

The audio flow controller (AFC) can control the flow of traffic between two I2S interfaces running at different clocks that are up to 100ppm apart.

The AFC implements high fidelity interpolation and decimation algorithms that compensate for clock differences.

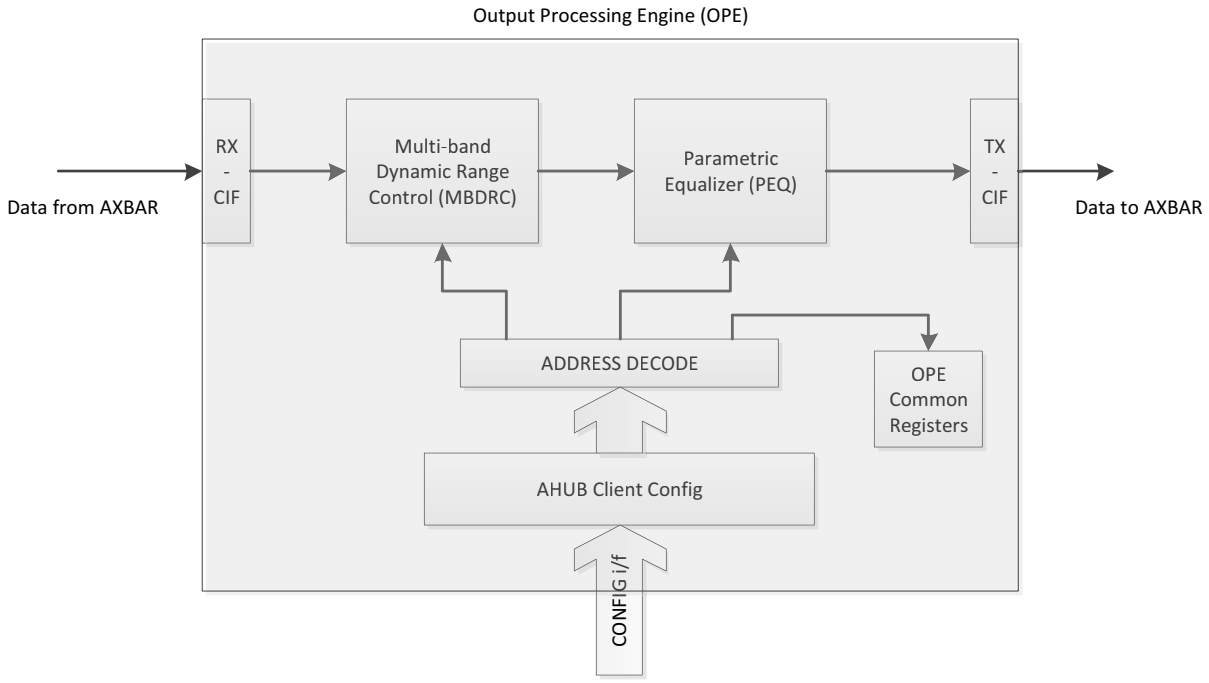
It can control traffic flow anywhere in the audio route, even to the inputs of AMX blocks.

The AFC can handle burst traffic from SFCs.

23.1.2.10 Output Processing Engine (OPE)

The Output Processing Engine (OPE) is one of the AHUB’s clients. The overall structure looks as shown below.

Figure 76: OPE Top-Level Block Diagram



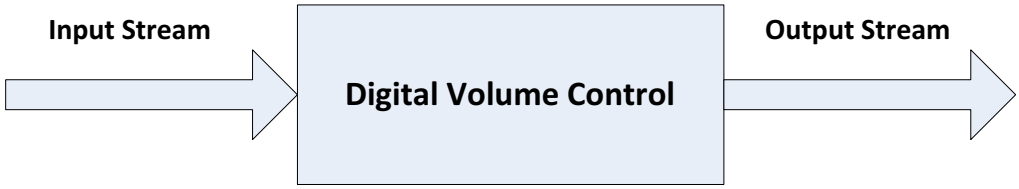
OPE Features

- Scalable Number of BiQuad Stages
- Supports stereo, 5p1 and 7p1 channels.
- Meets ultra-low power (ULP) audio requirements

23.1.2.11 Master Volume Control (MVC)

Digital volume control provides gain or attenuation to a digital signal path. The MVC digital volume control block can be used in input or output digital signal path. It can be used for per-stream volume control as well as master volume control. The following figure shows a simplified block diagram of digital volume control.

Figure 77: Simplified Digital Volume Control Block Diagram



The MVC block has one input and one output. The input digital stream can be mono- or multi-channel (up-to 7.1-channels) stream. The output digital stream has the same format as the input stream. Therefore, the sample rate, number of channels, number of bits per sample in the output stream are the same as those in the input stream. An independent mute/unmute control is also included in the MVC block. In addition, whenever the gain or mute setting is changed, the gain applied in digital volume control block is ramped up or down to the new setting. The parameters of the ramp, such as the duration and ramp-curve are programmable. In addition to linear ramp this design also provides hardware acceleration for the Windows audio curve type fade. The windows audio curve type fade is typically implemented using a programmable processor.

The MVC block also has an inbuilt peak meter detector. Peak meter detector is used as a feedback to the user. It is located after volume application and provides the peak output data for each channel. The duration used for the peak meter calculation is programmable

MVC Features

- Programmable range and steps for volume control
- Programmable Curve Ramp for volume control
- Independent mute/unmute controls
- Default gain values
- Peak meter detector

23.1.2.12 Audio Direct Memory Access (ADMA) Interface

The ADMA interface is a gateway in the AHUB for facilitating DMA transfers between memory and all of its clients. Basically it acts as a bridge to convert 32-bit full duplex AXI protocol to 1-bit DVS protocol with sufficient buffering for supporting seamless data transfers.

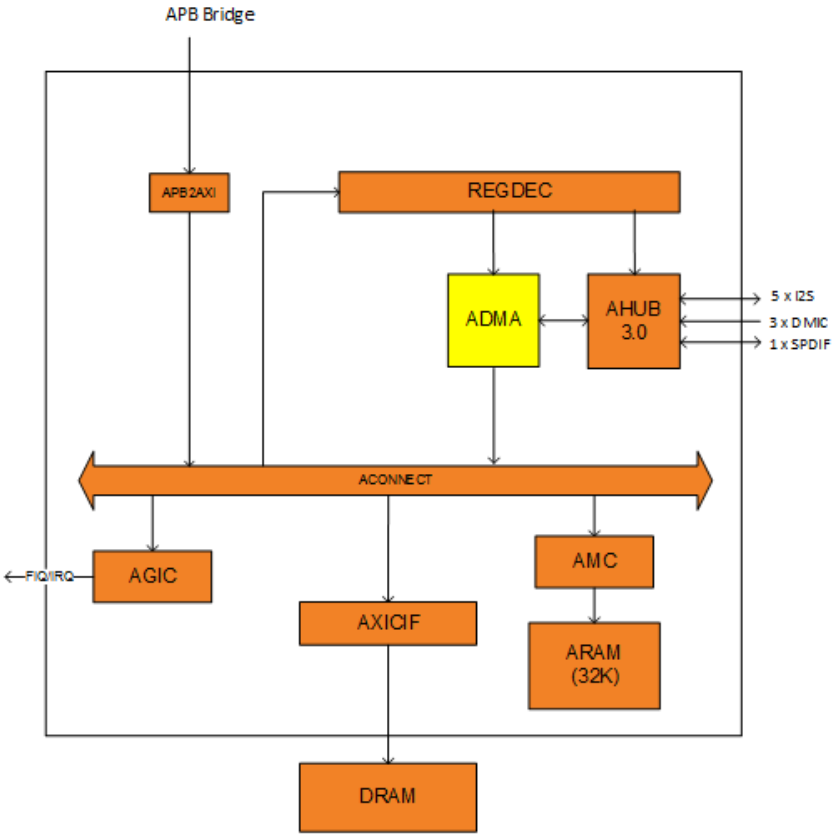
ADMA Interface Features

- 2 KB of buffering per direction for seamless DMA transfers.
- Dynamic or software configurable buffer allocation based on bandwidth and latency needs of each channel
- A maximum of 10 channels can be configured per direction for audio samples transfer between memory and AHUB clients.
- 32-bit full duplex AXI slave interface to communicate with ADMA for DMA transfers
- Up to 16 words of burst size
- APB slave interface for register programming
- 1-bit ACIF interface per channel to communicate with other AHUB clients through AXBAR
- Samples transfer through both DMA and register access (PIO mode)
- Supports packing (8 or 16 bits to 32 bits) in upstream direction and unpacking (32 bits to 8 or 16 bits) in downstream direction

23.1.3 ADMA

The ADMA caters the DMA needs of APE clients. It performs audio samples transfer among system memory, internal memory/ARAM, and AHUB through multiple unidirectional channels. A unidirectional channel is a point-to-point virtual connection for facilitating blocks of data movement with minimal CPU intervention. As depicted in the following figure, ADMA is plugged into the APE with the interfaces AXI interconnect for datapath, AHOST/REGDEC for programming interface, and AHUB for DMA requests originating from its internal clients.

Figure 78: The Sample Unit in the APE Block Diagram



23.1.3.1 ADMA Features

- Compliant with the industry standard AXI3 specifications for Data Path interface and internal standard HOST1X for programming interface.
- 22 channels where each channel can be configured for DMA transfer in any of 4 possible directions
  a. MEMORY-to-AHUB
  b. AHUB-to-MEMORY
  c. AHUB-to-AHUB
  d. MEMORY-to-MEMORY
- Flow controlled (Memory ←→ AHUB or AHUB ←→ AHUB) and non-flow controlled (Memory←→ Memory) DMA transfers

  DMA requests are made based on FIFO status of the requesters in case of flow controlled transfers
- Two AXI Master Interfaces.

  One of them is dedicated for transferring data to and from AHUB and the other one is hooked up to ACONNECT for the transfers to and from ARAM/DRAM
- Per channel pre-mask interrupt for indicating transfer completion to the AGIC. AGIC routes the post-mask interrupt to ADSP/CPU.
- 28-bit word aligned transfer size that can be configured up to 1 GB.
- 30-bit Word aligned source and target addresses
- Page or 4KB boundary alignment
- Per channel configurable burst size up to 16 words for all transfers

- Auto initialization of transfer size, source and target addresses on completion of programmed words transfer in continuous mode
- Scatter/Gather through Linked list for accessing non-contiguous memory regions

Software may use this feature in two scenarios:

1. When it requires to stream audio samples spread across the memory
2. When it requires less intervention from CPU. CPU need not attend the end of transfer interrupts from ADMA on a regular basis. It can occasionally write a bunch of descriptors into memory and attend other high priority tasks.

   Descriptor is a set of 4 registers (SOURCE_ADDR, TARGET_ADDR, TRANSFER_COUNT and NEXT_DESCRIPTOR_ADDR) located in consecutive 32-bit entries of DRAM/ARAM starting with SOURCE_ADDR.

   ADMA fetches next descriptor (SOURCE_ADDR, TARGET_ADDR, TRANSFER_COUNT and NEXT_DESCRIPTOR_ADDR) pointed to by the “NEXT_DESCRIPTOR_ADDR” of current descriptor for the next transfer while current transfer is in progress and these values are loaded at the end of the current transfer.

   Initially it is loaded by software while setting up the channel and then automatically loaded by hardware prior to end of each block transfer to enable seamless streaming. ADMA keeps fetching the descriptor from ARAM/DRAM pointed to by "NEXT_DESCRIPTOR_ADDR" until either software disables the channel or “NEXT_DESCRIPTOR_ADDR" register value of current descriptor is "0"

- Configurable address wrapping window in multiples of burst size for source and target in all modes
- Up to 8 buffers in all 3 transfer modes (linked list, once and continuous modes)
  - Source and target addresses are initialized on completion of all programmed buffers transfer and interrupt is generated on completion of every buffer transfer in continuous and linked list modes
  - Interrupt is generated on every buffer completion and channel would be disabled on completion of all buffers in once mode
- Up to 8 outstanding reads and writes
  - ADMA allows a maximum of 8 reads to source memory and that many writes to the target memory to be outstanding at any given point of time
  - ADMA issues read/write requests to ARAM/DRAM by mapping channel number to the “ARID/AWID” of AXI interface so that requests from different channels can pass each other and requests from same channel are handled in order or on first-come-first-serve basis en route to the target elements.
- Weighted Round Robin arbitration scheme for selecting read requests to AHUB and Memory
  - Each DMA channel can be programmed to a weight/request count so that it gets the grant for issuing that many requests back-to-back. This is applicable for all flow controlled (AHUB) and non-flow controlled transfers (ARAM/DRAM).
  - Separate WRR arbiters are used for AHUB and Memory Reads to enable un-interrupted full duplex transfer
- Configurable triggers for initiating DMA transfer.
  - Triggers for each channel are mapped to end of channel interrupt of all other DMA channels so that software can choose one of them to instruct the channel to initiate the transfer upon assertion of that particular interrupt
- Cut-through data transfers.
  - ADMA converts the returned data for the read requests issued to source element into write requests to the target element and forward them on AXI write channels in cut-through mode.

23.1.4 REGDEC

REGDEC provides the host interface for ADMA and AHUB in the APE. REGDEC provides only a direct register access path from ADSP/CPU to ADMA/AHUB. Command DMA and command synchronization features in AHOST are not supported by REGDEC.

REGDEC receives register requests targeted to ADMA and AHUB, decodes the requests, and forwards the requests to the corresponding REGDEC clients.

Features

- AXI3 interface to ACONNECT
- Host1x client interface to AHUB and ADMA
- Direct Register Access for AHUB and ADMA
- Frequency: 100MHz

23.1.5 ACONNECT

ACONNECT serves as the AXI interconnect backbone. There are AXI slave ports that connect to other engines, such as ADSP, ADMA, and AAS. AXI master ports connect to the ABRIDGE, AGIC and REGDEC. AXI to AMISC interface is APB.

The AXI receives memory read/write requests from the AXI masters and forwards to ABRIDGE/AMC/REGDEC/AMISC. It is responsible for transaction ordering, if any.

ACONNECT Features

- Frequency: 300 MHz
- Direct requests to slave target based on request address
- 64-bit AXI switch
- Compliant to transaction ordering defined in AXI3 and APB3 protocol specification. Transactions may be processed out-of-order by ACONNECT.

23.1.6 Audio Memory Controller (AMC)

Audio RAM provides a low latency local ARAM to DSP processor and other modules. Eight 32-bit EVP registers provide vector table at reset.

AMC Features

- AXI3 bus interface to ACONNECT
- 64-bit read and write data channels
- 64K ARAM
- Maximum Frequency: 300MHz
- 16 32-bit EVP registers

23.1.7 Audio Bridge (ABRIDGE)

ABRIDGE connects the internal AXI Interconnect/NIC (ACONNECT) to the Memory Controller via an AXICIF/MCCIF. ABRIDGE runs on the same first level clock as ACONNECT to reduce the number of clock crossings.

ABRIDGE Features

- AXI3
- MCCIF Version 2

23.1.8 APE Interrupt Controller (AGIC)

AGIC serves as the interrupt controller in the APE. It aggregates the interrupt lines from all the APE modules and distributes them to ADSP and the system. AGIC is an implementation of the ARM GIC-400 Generic Interrupt Controller, revision r0p1. Details of the GIC-400, including the technical reference manual, are available from ARM and may be downloaded from their website, http://arm.com.

GIC-400 supports 7 private peripheral interrupts (PPIs) per processor. For AGIC, PPIs are not used. GIC-400 supports lockable SPIs (LSPIs), AGIC does not use lockable SPIs feature. There are 56 SPI signals in AGIC. AGIC supports 2 SGIs for ADSP and

2 SGIs for CPU. SGIs are generated by writing to the Software Generated Interrupt Register, GICD_SGIR. ADSP can send 2 different interrupts to CPU and vice versa. AGIC has 2 CPU interfaces. CPU0 interface connects to system LIC which can then connect to the CPU. CPU1 interface connects to the ADSP.

23.1.8.1 Shared Peripheral Interrupts (SPI)

There are 2 interrupts line (FIQ/IRQ) per channel. Interrupts that are mapped to FIQ should use group 0. Interrupts that are mapped to IRQ should use group 1. Group 0 are recommended for higher priority interrupts.

Table 117: Interrupt Signals to GIC

Signal	ID	I/O	Edge/Level	Description
amisc_mbox_full[3:0]	32-35	I	Level	AMISC Mailbox Full Interrupt
amisc_mbox_empty[3:0]	36-39	I	Level	AMISC Mailbox Empty Interrupt
amisc_cpu_arb_sema[7:0]	40-47	I	Level	AMISC CPU Arbitrated Semaphore Interrupt
amisc_adsp_arb_sema[7:0]	48-55	I	Level	AMISC ADSP Arbitrated Semaphore Interrupt
adma_eot[21:0]	56-77	I	Level	ADMA Channel End of Transfer Interrupt
adsp_pmuirq	78	I	Level	ADSP/PTM Performance Monitoring Unit Interrupt
adsp_wdresetreq	79	I	Edge	ADSP Watchdog Timer Reset Request
adsp_l2ccintr	80	I	Level	ADSP L2 Cache Controller Interrupt
ahub_err	81	I	Level	AHUB Error Interrupt
amc_err	82	I	Level	AMC Error Interrupt
adma_err	83	I	Level	ADMA Error Interrupt
adsp_standbywfi	84	I	Level	ADSP Standby WFI. ADSP in idle mode. Waiting for Interrupt
adsp_standbywfe	85	I	Level	ADSP Standby WFE. ADSP in idle mode. Waiting for Event
adsp_ctiirq	86	I		
adsp_actmon	87	I	Level	Activity monitoring on ADSP. This interrupt is from AMISC
regdec_err (aas)	88	I	Level	This interrupt is from AAS

> ***Note:*** *Notice that ID assignment of SPI starts from 32. Asynchronous interrupt signals are synchronized in AGIC before sending to GIC-400. Signal adsp_wdresetreq is edge-triggered. The rest of APE interrupts are level-triggered.*

23.1.8.2 AGIC Features

- 32-bit AXI-4 compliant slave interface to ACONNECT
- Frequency: 300 MHz

23.1.8.3 Programming Sequence

Identifying the Supported Interrupts

This section discusses how to discover which the interrupts that are supported. Refer to Section 3.1.2 of GIC Architecture Specification.

1. Read GICD_TYPER. ITLinesNumber field identifies the number of implemented GICD_ISENABLERns.
2. Write to GICD_CTLR to disable forwarding of interrupts from distributor to the CPU interfaces
3. For each implemented GICD_ISENABLERn, starting with GICD_ISENABLER0:
   - Write 0xFFFFFFFF to the GICD_ISENABLERn.
   - Read the value of the GICD_ISENABLERn. Bits that read as 1 correspond to supported interrupt IDs.

Discovering permanently enabled interrupts

1. Write 0xFFFFFFFF to the GICD_ICENABLERn. This disables all interrupts that can be disabled.
2. Reads the value of the GICD_ICENABLERn. Bits that read as 1 correspond to interrupts that are permanently enabled.
3. Write 1 to any GICD_ISENABLERn bits corresponding to interrupts that must be re-enabled.

Routing an SPI interrupt to CPU/ADSP through FIQ using Group 1

1. Write 0xFFFFFFFF to the GICD_ICENABLERn. This disables all interrupts that can be disabled.
2. Enable Distributor Control Register

   Write 0x2 to Distributor Control Register (GICD_CTLR). Enable Grp1 interrupts.
3. Configure interrupt type (level or edge) by writing to Interrupt Configuration Registers, GICD_ICFGRn.
4. Write to interrupt group assignment by writing to the GICD_IGROUPR bit, if necessary. Set the intended interrupt to Group1.
5. Assign interrupt priority by writing to Interrupt Priority Registers, GICD_IPRIORITYRn, if necessary.
6. Write to GICD Priority Masked Interrupt GICD_PMR to set the interrupt mask level.
7. Write to Interrupt Processor Target Registers, GICD_ITARGETSRn to direct the interrupt to CPU.
8. Enable the interrupts by writing to Interrupt Set-Enable Register, GICD_ISENABLERn.
9. Enable CPU Interface Control Register (GICC_CTLR) of the destined CPU. AGIC is based on generic interrupt controller ARM IP GIC400. It serves as the APE interrupt controller, which detects, manages, and distributes interrupts. The GIC-400 complies to AMBA® AXI4 protocol and Version 2 of the ARM GIC Architecture Specification. The GIC-400 implements the GICv2 Security Extension. GIC does not support virtual extension. The feature is implemented in GIC-400 but will not be used or verified. Cortex-A9 doesn't support virtualization architecture.

   - FIQEn <= 1.

Routing an SPI interrupt to CPU/ADSP through FIQ using Group 0

1. Write 0xFFFFFFFF to the GICD_ICENABLERn. This disables all interrupts that can be disabled.
2. Enable Distributor Control Register

   - Write 0x1 to Distributor Control Register (GICD_CTLR). Enable Grp0 interrupts.
3. Configure interrupt type (level or edge) by writing to Interrupt Configuration Registers, GICD_ICFGRn.
4. Write to interrupt group assignment by writing to the GICD_IGROUPR bit, if necessary. Set the intended interrupt to Group0.
5. Assign interrupt priority by writing to Interrupt Priority Registers, GICD_IPRIORITYRn, if necessary.
6. Write to GICD Priority Masked Interrupt GICD_PMR to set the interrupt mask level.
7. Write to Interrupt Processor Target Registers, GICD_ITARGETSRn to direct the interrupt to CPU.
8. Enable the interrupts by writing to Interrupt Set-Enable Register, GICD_ISENABLERn.
9. Enable CPU Interface Control Register (GICC_CTLR) of the destined CPU. See ARM GIC Architecture Specification version 2, Tables 2-2 and 2-3 for IRQ and FIQ behavior.

   a. FIQEn <= 1.

   b. EnableGrp0 <= 1. Enable Group 0 interrupts using the FIQ signal. Group1 interrupts always using IRQ signal.

23.1.9 AMISC

This module contains ADSP configuration registers, synchronization registers, clock registers, and debug registers.

Features

- APB3 bus interface to ACONNECT

- 32-bit data width
- Frequency: 300MHz

23.1.10 APB-AXI Shim (AAS)

AAS is an APB2AXI shim which converts APB requests into AXI requests. The CPU accesses APE registers through the AAS over the APB bus. The AAS converts the requests to AXI and sends to ACONNECT which forwards the requests to the destination slaves.

The APE is not secured. It ignores the NS bit on the APB bus. Forward all requests to ACONNECT as secure.

23.1.10.1 Features

- Maximum Frequency: 300MHz
- Maximum of one outstanding APB request
- Support request
  - INCR 1: 32-bit read/write

23.2 Programming Guidelines

23.2.1 Common Programming Guidelines

23.2.1.1 System Level Programming

1. To start a use case: Enable modules from the tail end of the audio routing towards the source of the audio routing. For example, if the audio samples are fetched from memory, mixed with system tones in the mixer, and then played at the I2S, enable I2S first, followed by the mixer and then the ADMA Interface and ADMA.
2. To end a use case: Disable modules from the module nearest to the source towards the destination of the audio routing. In the use case example in point 1, disable ADMA first, followed by ADMA Interface, Mixer, and finally the I2S.
   a. The best practice while ending a use case is to disable a module using <module>_ENABLE register, wait for the <module>_STATUS_ENABLE_STATUS to become 0, and then move on to disabling the next module
   b. Alternatively, instead of polling for <module>_STATUS_ENABLE_STATUS to become 0, the software can choose to use the TX_DONE interrupt of every module. The “Using Interrupts” subsection explains this in detail.
   c. Besides the above options a. and b., for practical reasons, software can wait 20ms after getting a DMA TRANSFER_DONE interrupt (as DMA is the source in most audio routings), and then use the method described in option a. If a I2S/DMIC is the source of audio in AHUB, then software can wait 20ms after checking for I2S/DMIC ENABLE_STATUS to be 0 and then use the method described in option A.

23.2.1.2 Using Interrupts

All AHUB modules have RX_DONE and TX_DONE interrupts that can be used while ending a use case. The procedure to use these interrupts is:

1. Unmask the interrupt by writing 0 into <module>_XBAR_RX/TX_INT_MASK.
2. All the TX_DONE, RX_DONE interrupts are logically OR’ed and sent to the GIC as the ahub_err interrupt and hence the interrupt service routine for ahub_err interrupts needs to handle the TX_DONE/RX_DONE interrupts
   a. Check the fields of the AHUB_INTR_STATUS_0/1 register to find out which module(s) cause the ahub_err to be asserted
   b. In that module, check the fields of <module>_INT_STATUS register to find out which of the module’s interrupts was fired

23.2.1.3 Programming Module RAMs

Modules like SFC and Mixer have internal RAMs to contain filter coefficients, configurations, etc. The following are the programming guidelines for writing into and reading from the RAMs:

1. To write to a RAM location:
    a. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write the address of the RAM entry to be written to into the RAM_ADDR field
    b. Write the data to be written into the RAM location in the <module>_AHUBRAMCTL_<module>_DATA_x register
    c. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write 1 into the RW field
2. To write sequentially into a bunch of contiguous RAM locations:
    a. Write 1 into the <module>_AHUBRAMCTL_<module>_CTRL_x register field SEQ_ACCESS_EN
    b. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write the address of the RAM entry to be written to into the RAM_ADDR field
    c. Write 1 into the <module>_AHUBRAMCTL_<module>_CTRL_x register field ADDRESS_INIT_EN
    d. Write the data to be written into the RAM location in the <module>_AHUBRAMCTL_<module>_DATA_x register
    e. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write 1 into the RW field
    f. Repeat steps d and e for every word that needs to be written to the RAM (the address will increment automatically)
3. To read to a RAM location:
    a. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write the address of the RAM entry to be read from into the RAM_ADDR field
    b. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write 0 into the RW field
    c. Read the data from the <module>_AHUBRAMCTL_<module>_DATA_x register
4. To write sequentially into a bunch of contiguous RAM locations:
    a. Write 1 into the <module>_AHUBRAMCTL_<module>_CTRL_x register field SEQ_ACCESS_EN
    b. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write the address of the RAM entry to be read from into the RAM_ADDR field
    c. Write 1 into the <module>_AHUBRAMCTL_<module>_CTRL_x register field ADDRESS_INIT_EN
    d. In the <module>_AHUBRAMCTL_<module>_CTRL_x, write 0 into the RW field
    e. Read the data from the <module>_AHUBRAMCTL_<module>_DATA_x register
    f. Repeat steps d and e for every word that needs to be read from the RAM (the address will increment automatically)

23.2.1.4 AHUB Sub-modules Soft Reset Sequence

AHUB sub-modules support software reset. Soft reset will reset FSM and state registers and flush the ACIF FIFOs. However, the effect of software reset on the sub-module interrupt status register is not consistent across all the sub-modules. Some modules clear the interrupt status register on soft reset but others do not. Following sequence can be used to ensure a consistent behavior.

AHUB sub-module soft reset programming sequence:

- Disable all interrupts of a sub-module
- Handle and clear all pending interrupts
- Issue soft reset to the sub-module

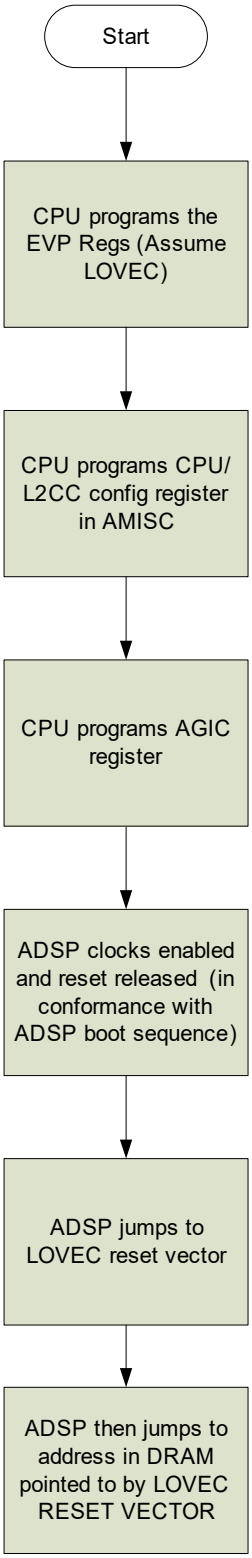
23.2.1.5 Boot Sequence

Cold Boot Sequence

The following steps are taken before bring ADSP out of reset:

- CPU enable clocks to APE through CAR module
- CPU programs CAR and de-asserts reset to ACONNECT, AAS, AMC, AGIC, AXIBRIDGE, AMISC, REGDEC, AHUB, ADMA
- CPU programs ACONNECT registers (if any) which configures ACONNECT
- CPU programs AGIC registers which sets the interrupt routing
- CPU configures AXIBRIDGE and AMC
- CPU configures ADSP and L2 cache controller through programming the configuration registers in AMISC
- CPU downloads the ADSP boot code to DRAM
- CPU programs EVP registers in AMC. Reset vector of EVP points to the start of the boot code.
- CPU de-asserts reset to the ADSP through the CAR module
- ADSP fetches the reset vector from the AMC and jumps to the boot code

Figure 79: ADSP Boot Flow



LP0/LP1 Entry

In LP0 state, DRAM is put in self-refresh mode and APE power rail (SOC) is off. In the LP1 state, DRAM is put in self-refresh mode and APE is power gated. There is no difference in the sequences to LP0 and LP1 states. Before entering LP0/LP1 state, APE states should be saved into DRAM. ADSP L1 and L2 are flushed.

LP0/LP1 Entry Sequence:

- CPU requests the ADSP to enter LP0. Request could come from mailbox.
- If the ADSP is not ready to enter LP0/LP1, it sends a “not ready” message to the CPU. Otherwise, continue.
- ADSP shuts off all activities in the AHUB and the ADMA. It makes sure there is no outstanding request in the ADMA. It is recommended that CPU does not access APE registers after asking the ADSP to enter LP0 and does not send more interrupts to the ADSP. It could unexpectedly interrupt the ADSP and divert it from the shut-down process.
- ADSP saves all states and restores information to DRAM
- ADSP flushes L1 and L2 caches
- ADSP disables interrupts to itself. It diverts interrupts to the CPU if appropriate.
- ADSP makes sure there are no outstanding activities.
- ADSP executes WFI/WFE. WFI is recommended because it makes sure there is no outstanding transaction in A9.
- CPU sees the interrupt from ADSP STANDBYWFI/STANDBYWFE signals. CPU reads the AGIC status register and makes sure STANDBYWFI or STANDBYWFE is asserted.
- CPU reads AMISC register to make sure PL310 CLKSTOPPED is asserted.
- CPU asserts reset and disables APE clocks through the CAR module.
- CPU clamps down and pulls down power to the APE.

LP0/LP1 Exit

LP0/LP1 exit boot sequence is the same as cold boot. To restore the ADSP context, the CPU sends a message to the ADSP, with pointers where the context information is saved.

- Execute ADSP cold boot sequence
- CPU sends restore commands, which have pointers to the saved data, to ADSP.

23.2.2 AAS Programming Guidelines

The AAS starts functioning right out of reset. No programming is needed.

When an error is asserted, read the INT_STATUS register to check for interrupt status. Write a ‘1’ to INT_STATUS_CLR to clear the corresponding interrupt status bit. When INT_STATUS is asserted, register ERROR_RESPONSE and ERROR_ADDRESS capture the information about the error status.

23.2.3 ACIF Programming Guidelines

The following programming guidelines are common for the ACIF registers in all AHUB modules. The programming guidelines of the respective modules will refer to this subsection when it comes to programming the ACIF registers.

1. Set the XBAR_CHANNELS field to the number of channels in the input or output stream of a module, depending on whether the ACIF in question is an RxCIF or TxCIF, respectively
2. Set the CLIENT_CHANNELS field to the number of channels of the stream as dealt with inside the module. This setting is different from XBAR_CHANNELS only when a conversion of the number of channels of an audio stream is desired.
3. Similarly, set the XBAR_BITS and CLIENT_BITS fields.
4. If XBAR_BITS and CLIENT_BITS do not match, the EXPAND and TRUNCATE fields are used to determine how to transition from XBAR_BITS to CLIENT_BITS (in the case of a RxCIF) or from CLIENT_BITS to XBAR_BITS (in the case of an TxCIF). Expanding always results in the actual data being shifted to the MSB bits and the rest filled with zeros, ones, or some random bits from an LFSR. TRUNCATE can be set to chopping or rounding.
5. For the ADMA interface, Iif packing of data in the TxCIF is desired, use PACK8_ENABLE or PACK16_ENABLE. Similarly, If unpacking of data in the RxCIF is desired, use UNPACK8_ENABLE or UNPACK16_ENABLE.

6. If XBAR_CHANNELS and the CLIENT_CHANNELS are different, the STEREO_CONV field is used to determine how a stereo stream is converted to a mono stream, and the MONO_CONV field is used to determine how a mono stream is converted to a stereo stream.

23.2.4 I2S Controller Programming Guidelines

I2S transmit and receive directions can be independently enabled/disabled and controlled.

1. Ensure that the I2S TX/RX is in a disabled state (after being used in a previous use case):

   Check if I2S_AXBAR_RX/TX_STATUS_ENABLE_STATUS = 0

2. Configure the I2S TX/RX path

   - Configure CIF parameters (see “ACIF Programming Guidelines” for more details):

   Populate I2S_XBAR_TX/RX_CIF_CTRL

3. Configure the I2S to operate in the mode of choice

   - Refer to the “Programming I2S to Operate in Various Modes” section about programming I2S in different modes.
   - Select the sampling rate indirectly by setting the channel bit count. Use the “I2S Sampling Rate Selection” section to find the correct value for this register:

     Configure I2S_TIMING_CHANNEL_BIT_CNT

4. Enable the I2S TX/RX:

   - Set I2S_AXBAR_TX/RX_ENABLE to 1.
   - In the case of I2S RX, ensure the enabling is complete before sending data to this module:

   Set I2S_AXBAR_RX_STATUS_ENABLE_STATUS

5. After using the module, disable it:

   Set I2S_AXBAR_TX/RX_ENABLE to 0.

23.2.4.1 I2S Sampling Rate Selection

The channel_bit_cnt can be calculated using the equation:

channel_bit_cnt = (frequency of bit_clk) / (C * required sampling rate) -1

Where C is 2 for LRCK modes and 1 for FSYNC modes.

If this calculation returns a fractional value, use the non-symmetry feature of the controller to attain the required sampling rate. In that case, the channel_bit_cnt value should be programmed with an integer that is closest to the fraction, but less than the fraction.

The table below contains some examples for common sampling rates with CLK_SOURCE_I2S = 24 MHz:

Sampling Rate	I2S_NEW_TIMING[12:00] (mark-space ratio: Left_channel : Right_channel)				
	CLK_DIVISOR=0 BIT_CLK=24 MHz	CLK_DIVISOR=1 BIT_CLK=12 MHz	CLK_DIVISOR=3 BIT_CLK=6 MHz	CLK_DIVISOR=5 BIT_CLK=4 MHz	CLK_DIVISOR=7 BIT_CLK=3 MHz
8 KHz	0x05DB	0x02ED	0x0176	0x00F9	0x10BB
	(1500:1500)	(750:750)	(375:375)	(250:250)	(187:188)
32 KHz	0x0176	0x10BB	Not supported	0x103E	Not supported
	(375:375)	(187:188)		(62:63)	
44.1 KHz	0x010F	0x0087	0x0043	0x002C	0x0021
	(272:272)	(136:136)	(68:68)	(45:45)	(34:34)
48 KHz	0x00F9	0x007C	0x103E	Not supported	Not supported
	(250:250)	(125:125)	(62:63)		

Sampling Rate	I2S_NEW_TIMING[12:00] (mark-space ratio: Left_channel : Right_channel)				
	CLK_DIVISOR=0 BIT_CLK=24 MHz	CLK_DIVISOR=1 BIT_CLK=12 MHz	CLK_DIVISOR=3 BIT_CLK=6 MHz	CLK_DIVISOR=5 BIT_CLK=4 MHz	CLK_DIVISOR=7 BIT_CLK=3 MHz
96 KHz	0x007C	0x103E	0x101F	Not supported	Not supported
	(125:125)	(62:63)	(31:32)		

Notes:

- *Ideally, any sampling rate can be generated, either accurately or approximately with any clk_src selected for I2S, if the clk_divisor and the channel_bit_cnt are programmed accordingly.*
- *The NON_SYM.EN feature is meant to create the sampling rates approximately by realizing an odd bit rate with a non-50:50 mark/space ratio. However, if the bit rate (2*channel_bit_cnt) itself is a fraction, the resultant sampling rate will not be accurate. The entries shown as not supported in the above table are attributed to such a deviation.*
- *When the channel_bit_cnt is less than 32, the bit_size should be programmed to be less than channel_bit_cnt.*

Programming I2S to Operate in Various Modes

After the module initialization (reset and clock programming), program the corresponding bits of the following registers.

Table 118: I2S Programming for Various Modes

Register	Basic	LJM	RJM	DSP	PCM	NW	TDM
I2S_CTRL_FRAME_FORMAT	0	0	0	1	1	1	1
I2S_CTRL_LRCK_POLARITY	0	1	1	1	1	1	1
I2S_CTRL_CHANNEL_BIT_CNT	Number of bit clocks in left channel	Number of bit clocks in left channel	Number of bit clocks in left channel	Number of bits in left channel + right channel	Number of bit clocks per frame	Number of bit clocks per frame	Number of bit clocks per frame
TX_DATA_OFFSET	1	0	CHANNEL_BIT_CNT -BITS_PER_SAMPLE	1	1: Short Fsync 0: Long Fsync	1	0/1/2
RX_DATA_OFFSET	1	0	CHANNEL_BIT_CNT – BITS_PER_SAMPLE	1	1:short Fsync 0:Long Fsync	1	0/1/2
I2S_CTRL_FSYNC_WIDTH	Number of bit clocks in left channel	Number of bit clocks in left channel	Number of bit clocks in left channel	1	0: Short Fsync 1: Long Fsync	0	Bits per sample (slot size)
HIGHZ_CTRL	0	0	0	0	2	2	0/1/2
EDGE_CTRL	0	0	0	0	1	1	1
I2S_SLOT_CTRL_TOTAL_SLOTS	0	0	0	1	0	3	0,1,...7
SLOT_ENABLES	0x01	0x01	0x01	0x03	0x01	*	**
* In NW mode, there can only be one active slot. So 0x01, 0x02, 0x04, 0x08 are allowed values. ** In TDM mode, there is no restriction on what/how many slots are enabled.							

The only restriction on the programming sequence is that all other registers must be programmed first before the Tx/Rx channels are enabled (which is done by setting XFER_EN_TX/XFER_EN_RX).

Programming I2S to Work in Loopback Mode

1. Set the I2S_CTRL_LPBK to 1.

The rest of the programming is the same as if I2S is not in the loopback mode.

23.2.5 DMIC Programming Guidelines

23.2.5.1 Minimum Required Guidelines

1. Ensure that the DMIC is in a disabled state (after being used in a previous use case):

   Check if DMIC_STATUS_ENABLE_STATUS = 0

   a. Configure the DMIC:

   b. Configure TxCIF parameters: Populate DMIC_XBAR_TX_CIF_CTRL registers

   c. Configure the OSR: Populate DMIC_CTRL_OSR register

   d. Enable the DMIC module: Set DMIC_ENABLE to 1

   e. After using the module, disable the module: Set DMIC_ENABLE to 0.

23.2.5.2 Optional Guidelines

While configuring the DMIC controller (step 2 of the section above) one can choose to configure the following optional features:

1. Configure whether to receive the left channel data followed by right channel data or the other way around:

   Use the DMIC_CTRL_LRSEL_POLARITY register field.

   a. Configure whether to receive data in both left and right channels or just the left channel or the right channel:

   Use the DMIC_CTRL_CHANNEL_SELECT register field

2. The trimmer value can be chosen to align data and clock lines from the DMIC device to the DMIC controller:

   Use the DMIC_CTRL_TRIMMER register field

23.2.6 S/PDIF Programming Guidelines

23.2.6.1 16-□20-□24-bit Modes

During transmission, the audio data is taken from the TX data FIFO, the channel status bit is taken from the TX channel status page buffer, and the user status bit is from the TX user FIFO. The preamble bits are generated by the hardware. The Valid bit is always zero.

During reception, the audio data is stored in the RX data FIFO, the channel status bit is stored in the RX channel status page buffer and the user status bit is stored in the RX user FIFO.

In addition to storing the audio data, the RX data FIFO also stores the preamble bits, channel status bit, user status bit and the valid bit. The reception of the channel bits start after the B-preamble is detected.

The audio-data sample has 16, 20, or 24 bits of data. Firmware has to transmit 16-, 20-, or 24-bit data and read 16-, 20-, or 24-bit data.

23.2.6.2 Raw Mode

During transmission, the audio data, the preamble bits, channel status bits, the user bits, and the valid bits are transmitted from the RX data FIFO. The firmware has to provide the correct preamble bits for the receiving device to synchronize with the transmitter. The firmware has to provide preambles according to the following table:

Preceding State in the Parity Bit: 0/1

Table 119: Preamble Bits

Symbol	Preamble Code	Channel Coding	Description
“B” (or “Z”)	0001/0001	11101000/00010111	Start of a block.
“M” (or “X”)	0010/0010	11100010/00011101	Start of sub-frame 1.
“W” (or “Y”)	0011/0011	11100100/00011011	Start of sub-frame 2.

During reception, the audio-data, the preamble bits, channel status bits, the user bits and the valid bits are stored in the RX data FIFO. The user bit is also stored in the RX user FIFO. The channel bit is also stored in the RX channel status page buffer. But the channel status bits are stored in the RX channel status page buffer only after the B-preamble is detected.

23.2.6.3 Module Reset

The application may initialize or reset the S/PDIF module via the clock and reset (CAR) control’s RST_DEVICES_L_0 register.

23.2.6.4 Clock Source/Divider Control

The SPDIFOUT clock source/divider can be selected by programming CAR’s CLK_SOURCE_SPDIF register. Refer to the S/PDIF registers below for the needed input frequency for the desire output audio sample rate.

The SPDIFIN clock source/divider can be selected by programming CAR’s CLK_SOURCE_SPDIF register. Refer to the SPDIF registers below for the needed oversample frequency.

23.2.6.5 Clock Enable Control

The application may enable the S/PDIF clocks by programming the CAR’s CLK_OUT_ENB_L_0 register.

23.2.6.6 Detecting Incoming Sampling Rate

The SPDIFIN is capable of detecting and locking onto the ‘spdifin’ data stream regardless of the sample rate. It achieves this by using an oversampling technique.

In addition to reading the sampling rate from the channel status information, one can also read the PERIOD field in the STROBE_CTRL_0 register to estimate the incoming sample rate. The last two digits of PERIOD indicate the fractional clock period.

Sample rate ≈ (‘spdifin’ clock frequency * 1000) / (PERIOD * 128)

If using the PERIOD method, it is better to read this field toward the end of the song or just before turning off the SPDIFIN. Although the hardware can lock onto the SPDIFIN data stream in as little as 2 subframes of time, the hardware is actually constantly re-adjusting itself (due to jitter, synchronization loss, etc. in the data stream) to provide the strobe point to the center of the biphase data stream. Over time, the strobe point will vary less and less because the hardware has already adjusted to all the variations seen in the data stream.

23.2.6.7 S/PDIF Transmit Data Flow

Make sure that the TX_FIFO is filled before enabling the transmission by setting the TX_EN field of CTRL_0 register. Never allow the TX_FIFO to go empty at any time during transmission. The interrupt is generated when the Tx Data FIFO empty count is greater than the Tx DATA attention value (TX_ATN_LVL field of the DATA_FIFO_CSR_0 register) if the Tx interrupt enable is set.

23.2.6.8 S/PDIF Receive Data Flow

To enable the reception of RX_FIFO, set the RX_EN field of the CTRL_0 register. Never allow the RX_FIFO to become full at any time during reception. The interrupt is generated when the Rx Data FIFO data count is greater than the Rx DATA attention value (RX_ATN_LVL field of the DATA_FIFO_CSR register), if the Rx interrupt enable bit is set.

23.2.6.9 Method of Filling/Retrieve Data from TX/RX User/Data FIFO

Use the DMA method by programming the APB DMA and APBIF registers. With this method, the FIFO attention levels need to be set in the APBIF just like in method 2. But instead of enabling interrupts when the FIFO attention level is reached, one should program the APB DMA registers. Now, every time the attention level is reached, a DMA request will be generated by the APBIF to the APB DMA and the APB DMA will perform the data moving task from/to APBIF FIFOs to the AHB bus.

23.2.7 Mixer Programming Guidelines

23.2.7.1 Guidelines for Programming Mixer Inputs

In general, the Mixer's inputs are all independent of each other. The programming guidelines are given for configuring and using a generic input. The guidelines are applicable for any of the Mixer's inputs.

1. Configure the CIF parameters.

   Populate the register MIXER_AXBAR_RX<no>_CIF_CTRL.
2. Make gain application related configurations (see Section 23.2.7.3: Programming Gain Application)
3. Make peak value related configuration

   Configure the register MIXER_AXBAR_RX<no>_PEAK_CTRL

23.2.7.2 Guidelines for Programming Mixer Outputs

1. Ensure the output that is intended to be used is in a disabled state (After being used in the previous use case)

   Check if MIXER_AXBAR_TX<no>_STATUS_ENABLE_STATUS is 0
2. Configure the adder associated with this output by enabling which inputs to be used with this adder

   Populate the register MIXER_AXBAR_TX<no>_ADDER_CONFIG
3. Ensure that the Mixer module is in an enabled state

   Check if MIXER_STATUS_ENABLE_STATUS is 1

   a. If not, enable the Mixer module

      Set MIXER_ENABLE to 1
4. Ensure the module is done initializing

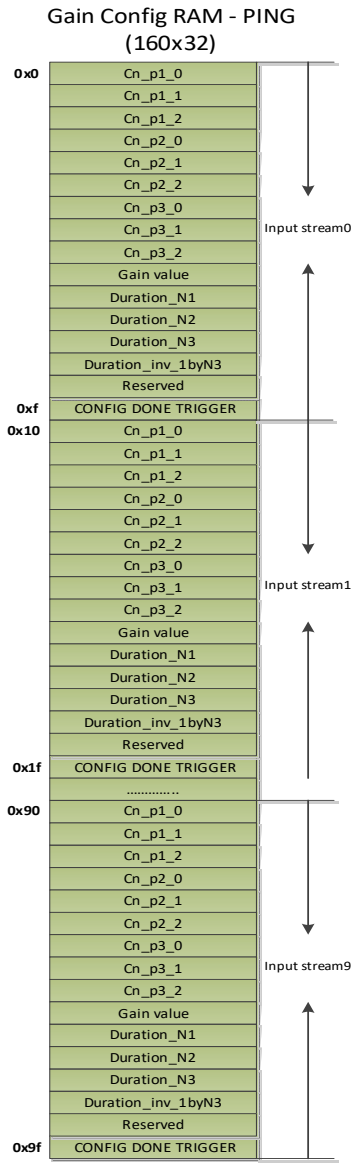
   Check if MIXER_STATUS_ENABLE_STATUS is 1
5. Enable the Mixer output

   Set MIXER_AXBAR_TX<no>_ENABLE to 1

23.2.7.3 Programming Gain Application

All gain-related configurations reside in a RAM as shown below.

Figure 80: Gain Configuration RAM



To write values into this RAM, one must use the AHUBRAMCTL_GAIN_CONFIG_RAM_CTRL, AHUBRAMCTL_GAIN_CONFIG_RAM_DATA. Refer to Section 23.2.1.3: Programming Module RAMs for programming guidelines on writing/reading RAM locations. The programming sequence for writing new gain parameters is:

1. Make sure the hardware is not in the middle of updating its internal variables with the old set of gain related parameters:

   Ensure MIXER_AXBAR_RX<no>_STATUS_CNFG_BUSY is 0

2. Program new gain value corresponding to other parameters if needed
3. Write to 0xf offset register to trigger the hardware to start using a new set of values.

   The ‘Duration_inv_1byN3’ should be calculated by software from the value of ‘Duration_N3’ (N3) as follows:

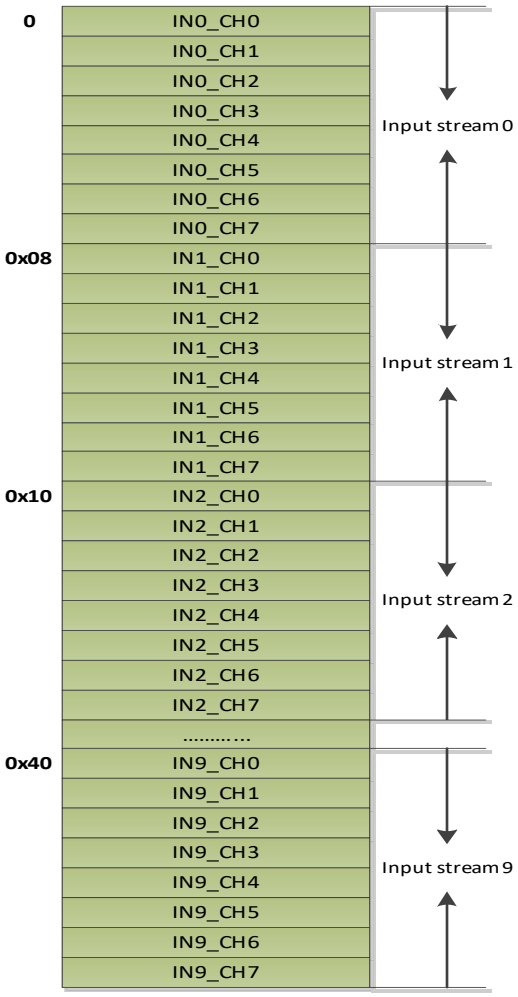
   Duration_inv_1byN3 = $\left(\frac{1}{N3}\right) * 2^{Prescalar} * 2^{31}$

   where, ‘Prescalar’ = 6.

23.2.7.4 Reading Peak Values

The peak value statuses are available in a RAM as shown below.

Figure 81: Peak Value RAM



To read a value from this RAM, one must use the registers AHUBRAMCTL_PEAKM_CTRL and AHUBRAMCTL_PEAKM_DATA. Refer to Section 23.2.1.3: Programming Module RAMs for programming guidelines on writing/reading RAM locations.

23.2.8 AMX Programming Guidelines

23.2.8.1 Mandatory Guidelines

1. Ensure that the AMX is in a disabled state (after being used in a previous use case)

   Check if AMX_STATUS_ENABLE_STATUS is 0

2. Configure the AMX

   a. Configure the RxCIF, TxCIF parameters (see Section 23.2.3: ACIF Programming Guidelines for more details)

      Populate AMX_XBAR_RX_CIF_CTRL0, AMX_XBAR_RX_CIF_CTRL1, AMX_XBAR_RX_CIF_CTRL2, AMX_XBAR_RX_CIF_CTRL3 and AMX_XBAR_TX_CIF_CTRL registers

   b. Enable the inputs that will be used for making the output frame:

      Use register fields AMX_CTRL_RX<no>_EN

3. Program the byte RAM map:

   Use AMX_AUDIORAMCTL_AMX_CTRL and AMX_AUDIORAMCTL_AMX_DATA registers. Refer to Section 23.2.1.3: Programming Module RAMs for programming guidelines on reading RAM locations.

4. Program the “wait-on” conditions for AMX:

   Configure the AMX_CTRL_RX_DEP, AMX_CTRL_MSTR_RX_NUM register fields

5. Configure which of the bytes in the output stream contain data

   Configure the AMX_OUT_BYTE_EN0 register for the LSB 32 bytes and the AMX_OUT_BYTE_EN1 register for the MSB 32 bytes

6. Enable the AMX module

   Set AMX_ENABLE to 1.

7. Ensure the module is done initializing before sending data to it

   Ensure AMX_STATUS_ENABLE_STATUS is 1

8. After using the module, disable the module

   Set AMX_ENABLE to 0.

23.2.8.2 Optional Guidelines

1. Configure AMX to stop accepting new input frames right after an input is disabled (otherwise AMX waits until all the continuous incoming frames are received and then disables)

   Set field(s) AMX_CTRL_RX<no>_DISABLE_MODE to FORCEFUL

   Note: Refer to the “Common Programming Guidelines” section for information on using some optional features in the AMX (and all other modules).

2. Configure Audio CIF control registers AMX_AUDIOCIF_CH0_CTRL, AMX_AUDIOCIF_CH1_CTRL, AMX_AUDIOCIF_CH2_CTRL, AMX_AUDIOCIF_CH3_CTRL, and AMX_AUDIOCIF_OUT_CTRL according to the guidelines given in Section 23.2.3: ACIF Programming Guidelines.

3. Configure the CH_DEP parameter field in the AMX_CTRL_0 register to determine whether to wait for all enabled channels to have data before transfer or start sending the data when any one of the input channels has data (WT_ON_ALL & WT_ON_ANY).

4. Configure the MSTR_CH_NUM register for designated master channel, so that if the data is available on a particular channel, the output can be sent.

5. Enable the AMX output by setting BYTE_EN in AMX_OUT_BYTE_EN0 and AMX_OUT_BYTE_EN1.

6. For Byte RAMCTL programming:

   - For programming RAM with the HW_ADR_EN bit cleared in the AMX_AUDIORAMCTL_AMX_CTRL register, set the RAM offset in the RAM_ADR field of the AMX_AUDIORAMCTL_AMX_CTRL register and set data in the DATA field in the AMX_AUDIORAMCTL_AMX_DATA register.
   - For programming RAM with the HW_ADR_EN bit set in the AMX_AUDIORAMCTL_AMX_CTRL register, set the RAM offset in the RAM_ADR field of the AMX_AUDIORAMCTL_AMX_CTRL register to 0 and set data in the DATA field in the AMX_AUDIORAMCTL_AMX_DATA register. Hardware auto-increments the RAM_ADR after each write.

23.2.9 ADX Programming Guidelines

23.2.9.1 Mandatory Guidelines

1. Ensure that the ADX is in a disabled state (after being used in a previous use case):

   Check if ADX_STATUS_ENABLE_STATUS is 0

2. Configure the ADX:

   a. Configure the RxCIF, TxCIF parameters (see Section 23.2.3: ACIF Programming Guidelines for more details)

Populate ADX_XBAR_TX_CIF_CTRL0, ADX_XBAR_TX_CIF_CTRL1, ADX_XBAR_TX_CIF_CTRL2, ADX_XBAR_TX_CIF_CTRL3 and ADX_XBAR_RX_CIF_CTRL registers

3. Enable the inputs that will be used for making the output frame:

   Use register fields ADX_CTRL_TX<no>_EN

4. Program the byte RAM map:

   Use ADX_AUDIORAMCTL_ADX_CTRL and ADX_AUDIORAMCTL_ADX_DATA registers. Refer to Section 23.2.1.3: Programming Module RAMs for programming guidelines on reading RAM locations.

5. Configure which of the bytes in the input stream contain data:

   Configure the ADX_IN_BYTE_EN0 register for the LSB 32 bytes and the ADX_IN_BYTE_EN1 register for the MSB 32 bytes

6. Enable the ADX module:

   Set ADX_ENABLE to 1

7. Ensure the module is done initializing.

   Ensure ADX_STATUS_ENABLE _STATUS is 1

8. After using the module, disable the module:

   Set ADX_ENABLE to 0.

23.2.9.2 Optional Guidelines

1. Configure ADX to stop generating new input frames right after an output is disabled (otherwise ADX waits until all the continuous incoming frames are received and then disables)

   Set field(s) ADX_CTRL_TX<no>_DISABLE_MODE to FORCEFUL

23.2.10 SFC Programming Guidelines

1. Ensure that the SFC is in a disabled state (after being used in a previous use case):

   Check if SFC_STATUS_ENABLE_STATUS = 0

2. Configure the SFC:

   a. Configure RxCIF, TxCIF parameters (see “ACIF Programming Guidelines” for more details):

      Populate SFC_AXBAR_RX_CIF_CTRL and SFC_AXBAR_TX_CIF_CTRL registers

   b. Configure input/output sampling frequencies:

      Populate SFC_AXBAR_RX_FREQ_FS_IN and SFC_AXBAR_TX_FREQ_FS_OUT registers

3. There are hardcoded coefficients for commonly used frequency conversions--see below. But for other frequency conversions, the software *must* program filter coefficients. Furthermore, optionally, even for the common use cases, the software may choose to program the filter coefficients. Note that the coefficients are in Q23 number format (1 sign bit and 23 fractional bits).

   a. Set SFC_COEF_RAM_0_COEF_RAM_EN to 1.

   b. Program the RAM via SFC_AHUBRAMCTL_SFC_CTRL_0 and SFC_AHUBRAMCTL_SFC_DATA_0. Refer to Section 23.2.1.3: Programming Module RAMs for how to program any RAM in an AHUB module.

4. Enable the SFC module:

   Set SFC_ENABLE to 1

5. Ensure the module initialization is done before sending data to SFC:

   Ensure SFC_STATUS_ENABLE_STATUS is 1

6. After using the module, disable the module:

   Set SFC_ENABLE to 0.

23.2.10.1 Commonly Used Frequency Conversions

- 8 KHz to 44.1 KHz
- 8 KHz to 48 KHz
- 16 KHz to 44.1 KHz
- 16 KHz to 48 KHz
- 44.1 KHz to 8 KHz
- 44.1 KHz to 16 KHz
- 48 KHz to 8 KHz
- 48 KHz to 16 KHz

23.2.11 AFC Programming Guidelines

1. Ensure that the AFC is in a disabled state (after being used in a previous use case):

   Check if AFC_STATUS_ENABLE_STATUS = 0

2. Configure the AFC:

   a. Configure RXCIF and TXCIF parameters (see Section 23.2.3: ACIF Programming Guidelines for more details)" Populate AFC_XBAR_RX_CIF_CTRL and AFC_XBAR_TX_CIF_CTRL registers

   b. Configure the maximum expected PPM difference between the two clock sources that AFC is used to control the flow between: Configure the AFC_CLK_PPM_DIFF register

   c. Configure the I2S FIFO thresholds and AFC TXCIF thresholds that are used by the AFC logic: Configure the AFC_CLK_PPM_DIFF register*

3. Enable the AFC module:

   Set AFC_ENABLE to 1

4. Ensure the module initialization is complete before sending data to AFC:

   Ensure AFC_STATUS_ENABLE_STATUS is 1

5. After using the module, disable the module:

   Set AFC_ENABLE to 0.

The following table shows the guidelines to follow for setting the threshold values under various conditions.

Note: *In the table, the numbers in parentheses are example values.*

Use Case	**AFC_THRESHOLDS_I2S_0**			**AFC_THRESHOLDS_AFC_0**			**Destination I2S FIFO Threshold (To be Programmed in I2S_AUDIOCIF_I2SRX_CTRL_0. FIFO_THRESHOLD)**
Actual Value (Typical Value)	HIGH_ THRESHOLD	START_ THRESHOLD	LOW_ THRESHOLD	HIGH_ THRESHOLD	START_ THRESHOLD	LOW_ THRESHOLD	
I2S → AFC → I2S	START_ THRESHOLD+ 1 (8)	START_ THRESHOLD (7)	START_ THRESHOLD - 1 (6)	START_ THRESHOLD+ 1 (8)	START_ THRESHOLD (7)	START_ THRESHOLD - 1 (6)	START_THRESHOLD + 1 (AFC_THRESHOLDS_ I2S_0) (8)
I2S → SFC → AFC → I2S	2*SRC-BURST + 1	1*SRC-BURST + 2	1*SRC-BURST	1*SRC-BURST + 1	1*SRC-BURST + 1	1*SRC-BURST + 1	START_THRESHOLD + 1 (AFC_THRESHOLDS_ I2S_0)

Use Case	AFC_THRESHOLDS_I2S_0			AFC_THRESHOLDS_AFC_0			Destination I2S FIFO Threshold (To be Programmed in I2S_AUDIOCIF_I2SRX_CTRL_0. FIFO_THRESHOLD)
I2S → AFC → AMX → I2S	START_THRESHOLD+1 (8)	START_THRESHOLD (7)	START_THRESHOLD - 1 (6)	START_THRESHOLD + 1 (8)	START_THRESHOLD (7)	START_THRESHOLD-1 (6)	Start Threshold + 1 (AFC_THRESHOLDS_I2S_0) (8)
I2S → SFC → AFC → AMX → I2S	2*SRC-BURST +1	1*SRC-BURST +2	1*SRC-BURST	2*SRC-BURST + 4	1*SRC-BURST + 4	(4)	START_THRESHOLD + 1 (AFC_THRESHOLDS_I2S_0)

23.2.12 OPE Programming Guidelines

1. Ensure that OPE is in a disabled state (after being used in a previous use case)

   Check if OPE_STATUS_ENABLE_STATUS = 0

2. Configure OPE:

   a. Configure RXCIF, TXCIF parameters (see Section 23.2.3: ACIF Programming Guidelines for more details)

      Populate OPE_XBAR_RX_CIF_CTRL and OPE_XBAR_TX_CIF_CTRL registers

3. Configure MBDRC and PEQ:
4. Enable OPE module:

   Set OPE_ENABLE to 1

5. Ensure the module initialization is complete before sending data to MBDRC:

   Ensure OPE_STATUS_ENABLE_STATUS is 1

6. After using the module, disable the module:

   Set OPE_ENABLE to 0.

If an output device is changing, or the coefficients and other settings need to be changed:

1. Disable OPE.
2. Wait for OPE_STATUS_ENABLE_STATUS = 0 (for datapath flushed out, or OPE_SOFT_RESET if immediate disable is needed)
3. Reprogram all coefficients and configuration registers as required.
4. Trigger OPE_SOFT_RESET.
5. Enable OPE as before.

Note: *Coefficient Data programmed takes care of the signs, i.e., hardware will always do add, so to implement a BiQuad operation b0*x[n] + b1*x[n-1] + b2*x[n-2] –a1*y[n-1] – a2*y[n-2], Software has to provide b0, b1, b2,-a1,-a2 as the coefficients.*

23.2.13 MVC Programming Guidelines

1. Ensure that the MVC is in a disabled state (after being used in a previous use case):

   Check if MVC_STATUS is 0

2. Configure the MVC

   a. Configure RXCIF and TXCIF parameters (see Section 23.2.3: ACIF Programming Guidelines for more details)

   b. Program MVC_CONFIG registers

   c. Program all Volume related settings (Target Volume, Duration, Polynomial Coefficients in Polynomial mode)

3. Enable MVC module:

   Set MVC_ENABLE to 1

4. Ensure the module initialization is complete before sending data to MVC:

   Ensure MVC_STATUS is 1

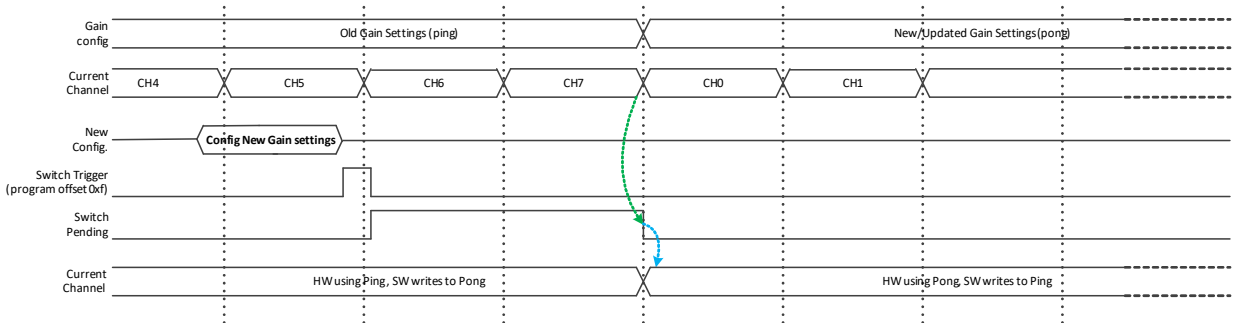
5. After using the module, disable the module:

   Set MVC_ENABLE to 0.

If software wants to change polynomial coefficients, duration parameters, or volume parameters (target volume and mute_unmute) on the fly while MVC is in enable state:

1. Check the status of switching: COEFF_SWITCH, DURATION_SWITCH and VOLUME_SWITCH
2. Program new parameters (For coefficient switching, hardware will handle whether that goes to Ping or Pong)
3. Set COEFF_SWITCH/DURATION_SWITCH/VOLUME_SWITCH to let hardware know new parameters are ready.

Figure 82: Programming Timeline for Updated Software Configuration



Note: *If software wants to program polynomial coefficients, duration parameters and volume parameters at the same time, then software should first program all the required fields for polynomial coefficients, duration parameters, and volume parameters, then set COEFF_SWITCH, DURATION_SWITCH, and VOLUME SWITCH together (they all belong to the same register).*

If an output device is changing, or the settings need to be changed:

1. Disable the MVC
2. Reprogram all registers
3. Trigger Soft Resets for MVC and CIFS
4. Enable – in same order above.

If the software is changing parameters while MVC is in disable state, following points should be kept in consideration:

1. If software wants to retain history between disable and enable stages, then when software disables the MVC, it should also set the INIT_VOL register to the last set Volume value by software. Examples:
   a. If software sets the volume to T1 and then disables the module, INIT_VOL should be set to T1.
   b. If software sets the volume to T1, then press Mute button and then disables the module, INIT_VOL should be set to T1
   c. If software sets the volume to T1 and then disables the module, INIT_VOL should be set to T1. Now if during the disable stage, if software changes the volume to T2, INIT_VOL should be set to T2.

      Exception: The only exception to the above case is if the last set volume was 0. If the last set volume was 0, then the INIT_VOL register should be set to the INIT_VOL during boot-up.
2. If software unmutes a channel while in the disabled stage, it needs to update the TARGET_VOL register to the last set volume value by software.

23.2.14 ADMA Programming Guidelines

1. Software should not attempt to write into registers of an active channel after the “TRANSFER_ENABLE” bit is set to “1”. The following register accesses are exceptions to this constraint.
   - “TRANSFER_PAUSE”
   - “SOFT_RESET”
2. ADMA channels to be involved in flow controlled transfers (AHUB-to-Memory and AHUB-to-AHUB) should be enabled prior to enabling the corresponding AHUB channels. For this, S/W should ensure “TRANSFER_ENABLED” status bit of ADMA channel is “high” before writing into “ENABLE_0_ENABLE” bit of corresponding AHUB channel.
3. Software should wait till “TRANSFER_ENABLED” status bit goes low and clear any pending interrupts for re-enabling a disabled channel.
4. “TRANSFER_COUNT” should be set to a non-zero value
5. Software should ensure “AHUB_FIFO_CTRL_0_TX_FIFO_SIZE” or “AHUB_FIFO_CTRL_0_RX_FIFO_SIZE” registers in an ADMA channel and “FIFO_CTRL_0_DMA_FIFO_SIZE” register of AHUB channel mapped to this ADMA channel programmed to a same value. ADMA and AHUB channel mapping is done through “CTRL_0_RX_REQUEST_SELECT” or “CTRL_0_TX_REQUEST_SELECT” registers in ADMA.
6. “TRANSFER_COUNT” status may not reflect the true value while transfer is in progress, so the channel should be “PAUSED” for getting the correct status.
7. Software may choose to terminate a transfer or disable a channel abruptly by clearing “TRANSFER_ENABLE” bit or wait until hardware disables it automatically up on normal completion of transfer.

23.3 Audio Processing Engine Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

23.3.1 Audio Crossbar (AXBAR) Registers

In the RX port registers, no more than one field can be 1 in a register because an RX port can receive from only one TX port.

23.3.1.1 AXBAR_PART_0_ADMAIF_RX*n*_0

There are 10 AXBAR_PART_0_ADMAIF_RX registers, one per RX port (*n* = 1 through 0xa).

Offset: 0x0 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.2 AXBAR_PART_0_I2S*i*_RX1_0

There are 5 AXBAR_PART_0_I2S_RX registers, five I2S channels per RX port (*i*= 1 through 5).

Offset: 0x40 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.3 AXBAR_PART_0_SFC*i*_RX1_0

There are 4 AXBAR_PART_0_SFC_RX registers, four SFC channels per RX port (*i* = 1 through 4).

Offset: 0x60 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.4 AXBAR_PART_0_MIXER1_RX*n*_0

There are 10 AXBAR_PART_0_MIXER_RX registers, one per RX port (*n* = 1 through 0xa).

Offset: 0x80 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.5 AXBAR_PART_0_SPDIF1_RX*n*_0

There are 2 AXBAR_PART_0_SPDIF_RX registers, one per RX port (*n* = 1 through 2).

Offset: 0xc0 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.6 AXBAR_PART_0_AFC*i*_RX1_0

There are 6 AXBAR_PART_0_AFC_RX registers, with six AFC channels per RX port (*i* = 1 through 6).

Offset: 0xd0 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.7 AXBAR_PART_0_OPE*i*_RX1_0

There are 2 AXBAR_PART_0_OPE_RX registers, with two OPE channels per RX port (*i* = 1 through 2).

Offset: 0x100 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.8 AXBAR_PART_0_MVC*i*_RX1_0

There are 2 AXBAR_PART_0_MVC_RX registers, with two MVC channels per RX port (*i* = 1 through 2).

Offset: 0x120 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.9 AXBAR_PART_0_AMX1_RX*n*_0

There are 4 AXBAR_PART_0_AMX1_RX registers, one per RX port (*n* = 1 through 4).

Offset: 0x140 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.10 AXBAR_PART_0_AMX2_RX*n*_0

There are 4 AXBAR_PART_0_AMX2_RX registers, one per RX port (*n* = 1 through 4).

Offset: 0x150 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.11 AXBAR_PART_0_ADX*i*_RX1_0

There are 2 AXBAR_PART_0_ADX_RX registers, with two ADX channels per RX port (*i*= 1 through 2).

Offset: 0x160 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxxxxx0000000000)

Bit	Reset	Description
27	0x0	SFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	SFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	SFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	SFC1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	I2S5_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	I2S4_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	I2S3_TX1: 0 = DISABLE 1 = ENABLE
17	0x0	I2S2_TX1: 0 = DISABLE 1 = ENABLE
16	0x0	I2S1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	ADMAIF_TX10: 0 = DISABLE 1 = ENABLE
8	0x0	ADMAIF_TX9: 0 = DISABLE 1 = ENABLE
7	0x0	ADMAIF_TX8: 0 = DISABLE 1 = ENABLE
6	0x0	ADMAIF_TX7: 0 = DISABLE 1 = ENABLE
5	0x0	ADMAIF_TX6: 0 = DISABLE 1 = ENABLE
4	0x0	ADMAIF_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	ADMAIF_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	ADMAIF_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	ADMAIF_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	ADMAIF_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.12 AXBAR_PART_1_ADMAIF_RX*n*_0

There are 10 AXBAR_PART_1_ADMAIF_RX registers, one per RX port (*n* = 1 through 0xa).

Offset: 0x200 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.13 AXBAR_PART_1_I2S*i*_RX1_0

There are 5 AXBAR_PART_1_I2S_RX registers, five I2S channels per RX port (*i*= 1 through 5).

Offset: 0x240 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.14 AXBAR_PART_1_SFC*m*_RX1_0

There are 4 AXBAR_PART_1_SFC_RX registers, four SFC channels per RX port (*m* = 1 through 4).

Offset: 0x260 + ([*m* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.15 AXBAR_PART_1_MIXER1_RX*a*_0

There are 10 AXBAR_PART_1_MIXER_RX registers, one per RX port (*a* = 1 through 0xa).

Offset: 0x280 + ([*a* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.16 AXBAR_PART_1_SPDIF1_RX*n*_0

There are 2 AXBAR_PART_1_SPDIF_RX registers, one per RX port (*n* = 1 through 2).

Offset: 0x2c0 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.17 AXBAR_PART_1_AFC*i*_RX1_0

There are 6 AXBAR_PART_1_AFC_RX registers, with six AFC channels per RX port (*i* = 1 through 6).

Offset: 0x2d0 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.18 AXBAR_PART_1_OPE*i*_RX1_0

There are 2 AXBAR_PART_1_OPE_RX registers, with two OPE channels per RX port (*i* = 1 through 2).

Offset: 0x300 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.19 AXBAR_PART_1_MVC*i*_RX1_0

There are 2 AXBAR_PART_1_MVC_RX registers, with two MVC channels per RX port (*i* = 1 through 2).

Offset: 0x320 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.20 AXBAR_PART_1_AMX1_RX*n*_0

There are 4 AXBAR_PART_1_AMX1_RX registers, one per RX port (*n* = 1 through 4).

Offset: 0x340 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.21 AXBAR_PART_1_AMX2_RX*n*_0

There are 4 AXBAR_PART_1_AMX2_RX registers, one per RX port (*n* = 1 through 4).

Offset: 0x350 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.22 AXBAR_PART_1_ADX*i*_RX1_0

There are 2 AXBAR_PART_1_ADX_RX registers, with two ADX channels per RX port (*i*= 1 through 2).

Offset: 0x360 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00xxxxxxxxxx00xxx00000)

Bit	Reset	Description
29	0x0	AFC6_TX1: 0 = DISABLE 1 = ENABLE
28	0x0	AFC5_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	AFC4_TX1: 0 = DISABLE 1 = ENABLE
26	0x0	AFC3_TX1: 0 = DISABLE 1 = ENABLE
25	0x0	AFC2_TX1: 0 = DISABLE 1 = ENABLE
24	0x0	AFC1_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	SPDIF1_TX2: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	SPDIF1_TX1: 0 = DISABLE 1 = ENABLE
9	0x0	AMX2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	AMX1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	MIXER1_TX5: 0 = DISABLE 1 = ENABLE
3	0x0	MIXER1_TX4: 0 = DISABLE 1 = ENABLE
2	0x0	MIXER1_TX3: 0 = DISABLE 1 = ENABLE
1	0x0	MIXER1_TX2: 0 = DISABLE 1 = ENABLE
0	0x0	MIXER1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.23 AXBAR_PART_2_ADMAIF_RX*n*_0

There are 10 AXBAR_PART_2_ADMAIF_RX registers, one per RX port (*n* = 1 through 0xa).

Offset: 0x400 + ([n * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.24 AXBAR_PART_2_I2S*i*_RX1_0

There are 5 AXBAR_PART_2_I2S_RX registers, five I2S channels per RX port (*i*= 1 through 5).

Offset: 0x440 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.25 AXBAR_PART_2_SFC*m*_RX1_0

There are 4 AXBAR_PART_2_SFC_RX registers, four SFC channels per RX port (*m* = 1 through 4).

Offset: 0x460 + ([*m* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.26 AXBAR_PART_2_MIXER1_RX*a*_0

There are 10 AXBAR_PART_2_MIXER_RX registers, one per RX port (*a* = 1 through 0xa).

Offset: 0x480 + ([*a* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.27 AXBAR_PART_2_SPDIF1_RX*n*_0

There are 2 AXBAR_PART_2_SPDIF_RX registers, one per RX port (*n* = 1 through 2).

Offset: 0x4c0 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.28 AXBAR_PART_2_AFC*i*_RX1_0

There are 6 AXBAR_PART_2_AFC_RX registers, with six AFC channels per RX port (*i* = 1 through 6).

Offset: 0x4d0 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.29 AXBAR_PART_2_OPE1_RX1_0

There are 2 AXBAR_PART_2_OPE_RX registers, with two OPE channels per RX port (*i* = 1 through 2).

Offset: 0x500 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.30 AXBAR_PART_2_MVC*i*_RX1_0

There are 2 AXBAR_PART_2_MVC_RX registers, with two MVC channels per RX port (*i* = 1 through 2).

Offset: 0x520 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.31 AXBAR_PART_2_AMX1_RX*n*_0

There are 4 AXBAR_PART_2_AMX1_RX registers, one per RX port (*n* = 1 through 4).

Offset: 0x540 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.32 AXBAR_PART_2_AMX2_RX*n*_0

There are 4 AXBAR_PART_2_AMX2_RX registers, one per RX port (*n* = 1 through 4).

Offset: 0x550 + ([*n* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.1.33 AXBAR_PART_2_ADX*i*_RX1_0

There are 2 AXBAR_PART_0_ADX_RX registers, with two ADX channels per RX port (*i*= 1 through 2).

Offset: 0x560 + ([*i* * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx000xx0000xx00xxx0xx00)

Bit	Reset	Description
31	0x0	ADX2_TX4: 0 = DISABLE 1 = ENABLE
30	0x0	ADX2_TX3: 0 = DISABLE 1 = ENABLE
29	0x0	ADX2_TX2: 0 = DISABLE 1 = ENABLE
28	0x0	ADX2_TX1: 0 = DISABLE 1 = ENABLE
27	0x0	ADX1_TX4: 0 = DISABLE 1 = ENABLE
26	0x0	ADX1_TX3: 0 = DISABLE 1 = ENABLE
25	0x0	ADX1_TX2: 0 = DISABLE 1 = ENABLE
24	0x0	ADX1_TX1: 0 = DISABLE 1 = ENABLE
20	0x0	DMIC3_TX1: 0 = DISABLE 1 = ENABLE
19	0x0	DMIC2_TX1: 0 = DISABLE 1 = ENABLE
18	0x0	DMIC1_TX1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x0	IQC2_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
14	0x0	IQC2_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
13	0x0	IQC1_TX2: Unused, reserved 0 = DISABLE 1 = ENABLE
12	0x0	IQC1_TX1: Unused, reserved 0 = DISABLE 1 = ENABLE
9	0x0	MVC2_TX1: 0 = DISABLE 1 = ENABLE
8	0x0	MVC1_TX1: 0 = DISABLE 1 = ENABLE
4	0x0	SPKPROT1_TX1: 0 = DISABLE 1 = ENABLE
1	0x0	OPE2_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	OPE1_TX1: 0 = DISABLE 1 = ENABLE

23.3.2 SFC Registers

23.3.2.1 SFC_AXBAR_RX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.2.2 SFC_AXBAR_RX_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	RX_DONE: 0 = FALSE 1 = TRUE

23.3.2.3 SFC_AXBAR_RX_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	RX_DONE: 0 = UNMASK 1 = MASK

23.3.2.4 SFC_AXBAR_RX_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	RX_DONE: This bit is auto cleared after an interrupt in generated 0 = FALSE 1 = TRUE

23.3.2.5 SFC_AXBAR_RX_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	RX_DONE: 0 = FALSE 1 = TRUE

23.3.2.6 SFC_AXBAR_RX_CIF_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00007700 (0bxx00000000000000x111x1110000x000)

Bit	Reset	Description
29:24	0x0	FIFO_THRESHOLD
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	Reset	Description
0	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.2.7 SFC_AXBAR_RX_FREQ_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	FS_IN: 0 = FS8 1 = FS11_025 2 = FS16 3 = FS22_05 4 = FS24 5 = FS32 6 = FS44_1 7 = FS48 8 = FS64 9 = FS88_2 10 = FS96 11 = FS176_4 12 = FS192

23.3.2.8 SFC_AXBAR_TX_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.2.9 SFC_AXBAR_TX_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = FALSE 1 = TRUE

23.3.2.10 SFC_AXBAR_TX_INT_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.2.11 SFC_AXBAR_TX_INT_SET_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	TX_DONE: This bit is auto cleared after an interrupt in generated 0 = FALSE 1 = TRUE

23.3.2.12 SFC_AXBAR_TX_INT_CLEAR_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	TX_DONE: 0 = FALSE 1 = TRUE

23.3.2.13 SFC_AXBAR_TX_CIF_CTRL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x000X7700 (0bxx0000000000xxxxx111x1110000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2
19:16	RO	X	CLIENT_CHANNELS: 0 = CH1 1 = CH2
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.2.14 SFC_AXBAR_TX_FREQ_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	FS_OUT: 0 = FS8 1 = FS11_025 2 = FS16 3 = FS22_05 4 = FS24 5 = FS32 6 = FS44_1 7 = FS48 8 = FS64 9 = FS88_2 10 = FS96 11 = FS176_4 12 = FS192

23.3.2.15 SFC_ENABLE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.2.16 SFC_SOFT_RESET_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.2.17 SFC_CG_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable,for first 2 channels and global/common logic. 0 = FALSE 1 = TRUE

23.3.2.18 SFC_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0xX0000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	SLCG_CLKEN: 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: 0 = FALSE 1 = TRUE

23.3.2.19 SFC_INT_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	TX_DONE: 0 = FALSE 1 = TRUE
0	X	RX_DONE: 0 = FALSE 1 = TRUE

23.3.3 I2S Registers

The I^2S Controller is designed to transport streaming audio-data between the system memory and an audio codec. The controller supports I^2S format, Left Justified Mode format, Right Justified Mode format, and DSP mode format, as defined in the Philips inter-IC-sound (I^2S) bus specification.

The controller has one transmit and one receive interface. The controller can be configured to operate either as a master or a slave.

23.3.3.1 I2S_AXBAR_RX_ENABLE_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.3.2 I2S_AXBAR_RX_SOFT_RESET_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
0	DISABLE	SOFT_RESET: 0 = DISABLE 1 = ENABLE

23.3.3.3 I2S_AXBAR_RX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x00000002

Bit	Reset	Description
2	FALSE	RXCIF_FIFO_FULL: 0 = FALSE 1 = TRUE
1	TRUE	RXCIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	FALSE	RX_ENABLED: 0 = FALSE 1 = TRUE

23.3.3.4 I2S_AXBAR_RX_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x00000000

Bit	Reset	Description
3	FALSE	RX_LOWER_WATERMARK: 0 = FALSE 1 = TRUE
2	FALSE	RX_NORMAL_WATERMARK: 0 = FALSE 1 = TRUE
1	FALSE	RXCIF_FIFO_UNDERRUN: 0 = FALSE 1 = TRUE
0	FALSE	RX_DONE: 0 = FALSE 1 = TRUE

23.3.3.5 I2S_AXBAR_RX_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000000f

Bit	Reset	Description
3	MASK	RX_LOWER_WATERMARK: 0 = UNMASK 1 = MASK
2	MASK	RX_NORMAL_WATERMARK: 0 = UNMASK 1 = MASK
1	MASK	RXCIF_FIFO_UNDERRUN: 0 = UNMASK 1 = MASK
0	MASK	RX_DONE: 0 = UNMASK 1 = MASK

23.3.3.6 I2S_AXBAR_RX_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
3	FALSE	RX_LOWER_WATERMARK: 0 = FALSE 1 = TRUE
2	FALSE	RX_NORMAL_WATERMARK: 0 = FALSE 1 = TRUE
1	FALSE	RXCIF_FIFO_UNDERRUN: 0 = FALSE 1 = TRUE
0	FALSE	RX_DONE: 0 = FALSE 1 = TRUE

23.3.3.7 I2S_AXBAR_RX_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
3	FALSE	RX_LOWER_WATERMARK: 0 = FALSE 1 = TRUE
2	FALSE	RX_NORMAL_WATERMARK: 0 = FALSE 1 = TRUE
1	FALSE	RXCIF_FIFO_UNDERRUN: 0 = FALSE 1 = TRUE
0	FALSE	RX_DONE: 0 = FALSE 1 = TRUE

23.3.3.8 I2S_AXBAR_RX_CIF_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00007700

Bit	Reset	Description
29:24	0x0	FIFO_THRESHOLD
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16

Bit	Reset	Description
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_MONO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3:2	0x0	FIFO_SIZE_DOWNSHIFT
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	0x0	MONO_STEREO_CONV: 0 = ZERO 1 = COPY

23.3.3.9 I2S_AXBAR_RX_CTRL_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
18:8	0x0	DATA_OFFSET: RX Data offset to FSYNC. 0 => No offset n => Data is offset by n bit CLK WRT FSYNC

Bit	Reset	Description
6:4	ZERO	MASK_BITS: Used for PCM mode. Set these mask bits to get the exact bit size in PCM mode. For example, to get 13 bit data, BIT_SIZE should be set to 3(BIT16) and MASK_BITS should be set to 3(THREE). HIGHZ_CTRL NOHIGHZ => Always drive sdataOut HIGHZ => Tristate SdataOut if a slot is not valid HIGHZ_ON_HALF_BIT_CLK => Tristate SdataOut after driving last bit data for half bitclk cycle 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN
2:1	NOHIGHZ	HIGHZ_CTRL: 0 = NOHIGHZ 1 = HIGHZ 2 = HIGHZ_ON_HALF_BIT_CLK
0	MSB_FIRST	BIT_ORDER: 0 = MSB_FIRST 1 = LSB_FIRST

23.3.3.10 I2S_AXBAR_RX_SLOT_CTRL_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
15:0	0x0	SLOT_ENABLES: Used in TDM mode to indicate which of the TDM slots contain data

23.3.3.11 I2S_AXBAR_RX_CLK_TRIM_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
4:0	0x0	TRIM_SEL: Clock trimmer for i2s RX clock segment

23.3.3.12 I2S_AXBAR_TX_ENABLE_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.3.13 I2S_AXBAR_TX_SOFT_RESET_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
0	DISABLE	SOFT_RESET: 0 = DISABLE 1 = ENABLE

23.3.3.14 I2S_AXBAR_TX_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x00000002

Bit	Reset	Description
2	FALSE	TXCIF_FIFO_FULL: 0 = FALSE 1 = TRUE
1	TRUE	TXCIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	FALSE	TX_ENABLED: 0 = FALSE 1 = TRUE

23.3.3.15 I2S_AXBAR_TX_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x00000000

Bit	Reset	Description
3	FALSE	TX_UPPER_WATERMARK: 0 = FALSE 1 = TRUE
2	FALSE	TX_NORMAL_WATERMARK: 0 = FALSE 1 = TRUE
1	FALSE	TXCIF_FIFO_OVERRUN: 0 = FALSE 1 = TRUE
0	FALSE	TX_DONE: 0 = FALSE 1 = TRUE

23.3.3.16 I2S_AXBAR_TX_INT_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x0000000f

Bit	Reset	Description
3	MASK	TX_UPPER_WATERMARK: 0 = UNMASK 1 = MASK
2	MASK	TX_NORMAL_WATERMARK: 0 = UNMASK 1 = MASK
1	MASK	TXCIF_FIFO_OVERRUN: 0 = UNMASK 1 = MASK
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.3.17 I2S_AXBAR_TX_INT_SET_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
3	FALSE	TX_UPPER_WATERMARK: 0 = FALSE 1 = TRUE
2	FALSE	TX_NORMAL_WATERMARK: 0 = FALSE 1 = TRUE
1	FALSE	TXCIF_FIFO_OVERRUN: 0 = FALSE 1 = TRUE
0	FALSE	TX_DONE: 0 = FALSE 1 = TRUE

23.3.3.18 I2S_AXBAR_TX_INT_CLEAR_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
3	FALSE	TX_UPPER_WATERMARK: 0 = FALSE 1 = TRUE
2	FALSE	TX_NORMAL_WATERMARK: 0 = FALSE 1 = TRUE
1	FALSE	TXCIF_FIFO_OVERRUN: 0 = FALSE 1 = TRUE
0	FALSE	TX_DONE: 0 = FALSE 1 = TRUE

23.3.3.19 I2S_AXBAR_TX_CIF_CTRL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00007700

Bit	R/W	Reset	Description
29:24	RO	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16

Bit	R/W	Reset	Description
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_MONO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3:2	RW	0x0	FIFO_SIZE_DOWNSHIFT
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_STEREO_CONV: 0 = ZERO 1 = COPY

23.3.3.20 I2S_AXBAR_TX_CTRL_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
18:8	0x0	DATA_OFFSET: RX Data offset to FSYNC. 0 => No offset n => Data is offset by n bit CLK WRT FSYNC
6:4	ZERO	MASK_BITS: Used for PCM mode. Set these mask bits to get the exact bit size in PCM mode. For example, to get 13 bit data, BIT_SIZE should be set to 3(BIT16) and MASK_BITS should be set to 3(THREE). 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN
0	MSB_FIRST	BIT_ORDER: 0 = MSB_FIRST 1 = LSB_FIRST

23.3.3.21 I2S_AXBAR_TX_SLOT_CTRL_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
15:0	0x0	SLOT_ENABLES: Used in TDM mode to indicate which of the TDM slots contain data

23.3.3.22 I2S_AXBAR_TX_CLK_TRIM_0

Offset: 0x6c | Read/Write: R/W | Reset: 0x00000000

Bit	Reset	Description
12:8	0x0	TRIM_SEL_MASTER: Clock trimmer for pad->Rx pad macro flops in slave mode
4:0	0x0	TRIM_SEL_SLAVE: Clock trimmer for pad->Rx pad macro flops in slave mode

23.3.3.23 I2S_ENABLE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.3.24 I2S_SOFT_RESET_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	SOFT_RESET: 0 = DISABLE 1 = ENABLE

23.3.3.25 I2S_CG_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_ENABLE: 0 = FALSE 1 = TRUE

23.3.3.26 I2S_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0x0000XX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12	X	SLCG_CLKEN: 0 = FALSE 1 = TRUE
10	X	RXCIF_FIFO_FULL: 0 = FALSE 1 = TRUE
9	X	RXCIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
8	X	RX_ENABLED: 0 = FALSE 1 = TRUE
2	X	TXCIF_FIFO_FULL: 0 = FALSE 1 = TRUE
1	X	TXCIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	TX_ENABLED: 0 = FALSE 1 = TRUE

23.3.3.27 I2S_INT_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	RXCIF_FIFO_UNDERRUN: 0 = FALSE 1 = TRUE
8	X	RX_DONE: 0 = FALSE 1 = TRUE
1	X	TXCIF_FIFO_OVERRUN: 0 = FALSE 1 = TRUE
0	X	TX_DONE: 0 = FALSE 1 = TRUE

23.3.3.28 I2S_CTRL_0

The I2S control register is used to configure bit formats (I2S, LJM, RJM, DSP, PCM, NW, TDM), bit sizes (8, 16, 20, 24 and 32), FIFO formats, Master/slave selection, LR polarity, and error interrupt enables.

I2S Control Register

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxx00xxxx000x000xxxxx000)

Bit	Reset	Description
31:24	0x0	FSYNC_WIDTH: In number of bit clocks.
20	0x0	EDGE_CTRL: Indicates on which edge to drive data, and on which edge to sample data 0 = Drive data on negative edge and sample data on positive edge 1 = Drive data on positive edge and sample data on negative edge 0 = POS_EDGE 1 = NEG_EDGE
19	0x0	PIPE_MACRO_EN: Enable pipe macro stage (1 stage in Tx direction and 1 in Rx direction). 0 = DISABLE 1 = ENABLE
14:12	0x0	FRAME_FORMAT: Frame format. 0: BASIC, LJM, RJM modes 1: DSP, PCM, NW, TDM modes 0 = LRCK_MODE 1 = FSYNC_MODE
10	0x0	MASTER: Controller Master/Slave mode selection. 0: Do not drive LRCK and clock. 1: Drive LRCK and clock 0 = DISABLE 1 = ENABLE
9	0x0	LRCK_POLARITY: Left/Right Control Polarity. 0= Left channel when LRCK is low 1= Left channel when LRCK is high 0 = LOW 1 = HIGH
8	0x0	LPBK: Tx->Rx Loop Back Enable 0 = DISABLE 1 = ENABLE
2:0	0x0	BIT_SIZE: Bits per sample. 0 = BIT_SIZE_RSVD 1 = BIT_SIZE_8 2 = BIT_SIZE_12 3 = BIT_SIZE_16 4 = BIT_SIZE_20 5 = BIT_SIZE_24 6 = BIT_SIZE_28 7 = BIT_SIZE_32

23.3.3.29 I2S_TIMING_0

The NON_SYM.EN bit can be used to program the I2S Controller to output a non-50:50 mark-space ratio on to the I2S bus. When the NON_SYM.EN bit is enabled, the Controller sends out exactly the programmed number of clock cycles on the left channel and one bit clock greater on the right channel. This is used in Basic, LJM, and RJM modes.

When NON_SYM.EN is enabled, the channel-bit-count should be programmed with the number of bit clocks required in the left channel. This will ensure that the non-50:50 mark-space ratio is achieved with R_BCLK = L_BCLK + 1.

For PCM/NW/TDM modes, channel_bit_cnt can be calculated using the following equation:

```
Channel_bit_cnt = (frequency of bit_clk) / (2 * required sampling rate) - 1;
```

If this calculation returns a fractional value, the non-symmetry feature of the controller should be used to attain the required sampling rate.

I2S Timing Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxx0x00000011111)

Bit	Reset	Description
12	0x0	NON_SYM_EN: Enables non-symmetry mode. 0 = DISABLE 1 = ENABLE
10:0	0x1f	CHANNEL_BIT_CNT: I2S, LJM, RJM mode: Number of bit clocks in left or right channel. DSP mode: Number of bit clocks in LEFT+RIGHT channel TDM/NW/PCM mode: Number of bit clocks in the frame

23.3.3.30 I2S_SLOT_CTRL_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	TOTAL_SLOTS: Number of slots when TDM mode is used.

23.3.3.31 I2S_CLK_TRIM_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000xxx00000)

Bit	Reset	Description
12:8	0x0	SCLK_TRIM_SEL: Clock trimmer for master mode i2s clock output to pad
4:0	0x0	CORE_TRIM_SEL: Clock trimmer for core i2s clock segment

23.3.4 SPDIF Registers

The S/PDIF consists of the following two major modules:

- The SPDIFOUT sub-module, which sends data to the "spdifout" port in IEC 60958-3 biphase-mark code format.
- The SPDIFIN sub-module, which retrieves data to the "spdifin" port in IEC 60958-3 biphase-mark code format.

23.3.4.1 SPDIF_CTRL_0

SPDIF Control Register

Note: *Changing the state of TC_EN, TU_EN, LBK_EN, PACK, BIT_MODE while RX_EN and/or TX_EN and/or RX_BSY and/or TX_BSY is set can cause unexpected behavior and therefore shall not be attempted.*

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b000000xxxxxxxxxx00000xxx0xxxxxxx)

Bit	Reset	Description
31	0x0	FLOWCTL_EN: 1=Enable flow control 0 = DISABLE 1 = ENABLE
30	0x0	CAP_LC: 1=start capturing from left channel,0=start capturing from right channel. 0 = RIGHT_CH 1 = LEFT_CH
29	0x0	RX_EN: SPDIF receiver (RX): 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
28	0x0	TX_EN: SPDIF transmitter (TX): 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	TC_EN: Transmit Channel status: 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
26	0x0	TU_EN: Transmit user Data: 0 = DISABLE 1 = ENABLE
15	0x0	LBK_EN: Loopback test mode: 1=Enable internal loopback, 0=Normal mode. 0 = DISABLE 1 = ENABLE
14	0x0	PACK: Pack data mode: 1=Packeted left/right channel data into a single word, 0=Single data (16 bits need to be padded to match the interface data bit size) 0 = DISABLE 1 = ENABLE
13:12	0x0	BIT_MODE: 00=16-bit data. 01=20-bit data. 10=24-bit data. 11=raw data. This value should match ACIF CLIENT_BITS. 0 = MODE16BIT 1 = MODE20BIT 2 = MODE24BIT 3 = MODERAW
11	0x0	CG_EN: 1=Enable second level clock gating 0 = DISABLE 1 = ENABLE
7	0x0	SOFT_RESET: This bit is Auto Cleared. Resets I2S logic including CIFs and flow control. Configuration registers are not reset by soft reset. 0 = DISABLE 1 = ENABLE

23.3.4.2 SPDIF_STROBE_CTRL_0

SPDIF Data Strobe Control Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxx0xx00000xx000000)

Bit	R/W	Reset	Description
23:16	RO	X	PERIOD: Indicates the approximate number of detected SPDIFIN clocks within a biphase period.
15	RW	0x0	STROBE: SPDIFIN Data Strobe Mode 1=Manual-locked strobe 0=Auto-locked strobe (default)
12:8	RW	0x0	DATA_STROBES: Manual data strobe time within the biphase clock period (in terms of the number of oversampling clocks)
5:0	RW	0x0	CLOCK_PERIOD: Manual SPDIFIN biphase clock period (in terms of the number of over-sampling or 'spdifin' clocks)

23.3.4.3 SPDIF_AUDIOCIF_TXDATA_CTRL_0

An SPDIF module has four AUDIOCIF interfaces for TX data, RX data, TX user data, and RX user data. CIF RX port for SPDIF TX DATA

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF

Bit	R/W	Reset	Description
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.4.4 SPDIF_AUDIOCIF_RXDATA_CTRL_0

CIF TX port for SPDIF RX DATA

Offset: 0xc │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.4.5 SPDIF_AUDIOCIF_TXUSER_CTRL_0

CIF RX Port for SPDIF USER Data

Offset: 0x10 | Read/Write: R/W | Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16

Bit	R/W	Reset	Description
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.4.6 SPDIF_AUDIOCIF_RXUSER_CTRL_0

CIF TX Port for SPDIF USER Data

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x0000110X (0bxx00000000000000x001x00100000x00)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16

Bit	R/W	Reset	Description
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.4.7 SPDIF_CH_STA_RX_A_0

This 6-word receive channel data page buffer holds a block (192 frames) of channel status information. The order of receive is from LSB bit to MSB bit, and from CH_STA_RX_A to CH_STA_RX_F and back to CH_STA_RX_A.

Note: *Only channel status bits from channel A (subframe 1) will be saved into this page buffer.*

SPDIF Channel Status Rx Page Buffer Register

Offset: 0x18 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C31_RX_C0: Channel status bits [31:0]; one bit per audio sample

23.3.4.8 SPDIF_CH_STA_RX_B_0

Offset: 0x1c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C63_RX_C32: Channel status bits [63:32]; one bit per audio sample

23.3.4.9 SPDIF_CH_STA_RX_C_0

Offset: 0x20 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C95_RX_C64: Channel status bits [95:64]; one bit per audio sample

23.3.4.10 SPDIF_CH_STA_RX_D_0

Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C127_RX_C96: Channel status bits [127:96]; one bit per audio sample

23.3.4.11 SPDIF_CH_STA_RX_E_0

Offset: 0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C159_RX_C128: Channel status bits [159:128]; one bit per audio sample

23.3.4.12 SPDIF_CH_STA_RX_F_0

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C191_RX_C160: Channel status bits [191:160]; one bit per audio sample

23.3.4.13 SPDIF_CH_STA_TX_A_0

This 6-word transmit channel data page buffer holds a block (192 frames) of channel status information. The order of transmission is from LSB bit to MSB bit, and from CH_STA_TX_A to CH_STA_TX_F and back to CH_STA_TX_A.

Note: *The channel status data from this page buffer will be used only when PACK=1, or when BIT_MODE=00/01/10.*
Each channel status bit will be sent out to "both" subframes.

SPDIF Channel Status Tx Page Buffer Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C31_TX_C0: Channel status bits [31:0]; one bit per audio sample

23.3.4.14 SPDIF_CH_STA_TX_B_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C63_TX_C32: Channel status bits [63:32]; one bit per audio sample

23.3.4.15 SPDIF_CH_STA_TX_C_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C95_TX_C64: Channel status bits [95:64]; one bit per audio sample

23.3.4.16 SPDIF_CH_STA_TX_D_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C127_TX_C96: Channel status bits [127:96]; one bit per audio sample

23.3.4.17 SPDIF_CH_STA_TX_E_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C159_TX_C128: Channel status bits [159:128]; one bit per audio sample

23.3.4.18 SPDIF_CH_STA_TX_F_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C191_TX_C160: Channel status bits [191:160]; one bit per audio sample

23.3.4.19 SPDIF_FLOWCTL_CTRL_0

This register controls the flow control mechanism and is valid only with the TX mode. The flow control is only useful when the input stream to I2S/SPDIF is already real-time spaced such as a live stream from Baseband through another I2S_RX. If the input stream is from DMA directly, the flow control should be disabled.

Offset: 0x70 | Read/Write: R/W | Reset: 0x80000000 (0b10xxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
31	QUAD	FILTER: to use linear or quadratic filter. Tegra K1 devices support quadratic filter only. 0 = LINEAR 1 = QUAD
30	DISABLE	DUMMY_WR_EN: Insert a dummy frame when enabled 0 = DISABLE 1 = ENABLE
11:8	0x0	START_THRESHOLD: Flow control should wait for FIFO filled N-1, 0 means actual threshold 1
7:4	0x0	HIGH_THRESHOLD: High-threshold for HIGH state N-1, 0 means actual threshold 1
3:0	0x0	LOW_THRESHOLD: Low-threshold for LOW state N-1, 0 means actual threshold 1

23.3.4.20 SPDIF_TX_STEP_0

Offset: 0x74 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	STEP_SIZE: This is a compensation value for the clock difference between input and output. To catch up the clock difference quickly, this value is supposed to be bigger than clock difference in terms of STEP_SIZE, but to avoid the sound quality degradation it should not be much greater than the clock difference. For example, the input frame sampling rate is 44.095 kHz, the output frame sampling rate is 44.100 kHz, and the clock error rate is (44100 - 44095) / 44100 = 0.00011.3 The step size used in the linear interpolation is 0x8000 (32768) step_size = 0.000113 * 32768 = 3.7. A threshold of 4 is recommended. If the value is less than 3 in the above example, the adjusted step size will not be able to catch up. The flow control FIFO will eventually become empty, leading to either NULL or replication of the last sample.

23.3.4.21 SPDIF_FLOW_STATUS_0

When the flow controller is used, the step size should be properly chosen depending on the clock frequency difference. The step size should be small enough not to degrade sound quality, but big enough to converge the FIFO depth. If the step size is not big enough, the FIFO depth will either increase or decrease, which will cause FIFO overflow/underflow eventually.

The flow controller monitor and counter can be used to fine-tune the step size. When the flow controller is enabled and a proper step size is chosen, the FIFO depth should be in [threshold -1 : threshold + 1].

When the flow controller monitor is enabled and the FIFO depth is deviated from the range, either FLOW_OVERFLOW or FLOW_UNDERFLOW will be set. Also, the interrupt signal will be asserted, if MONITOR_INT_EN is enabled. MONITOR_CLR clears the FLOW_OVERFLOW/UNDERFLOW as well as the interrupt signal to APBIF.

When the flow controller counter is enabled, it records how many samples are processed and how many times the step size is applied to either lower or raise the time step. In case of stereo samples, it counts as 1.

- FLOW_NORMAL - the total number of samples (only count left channel samples).
- FLOW_OVER - the number of samples with adding STEP_SIZE to the time step.
- FLOW_UNDER - the number of samples with subtracting STEP_SIZE from the time step.

Flow Controller Monitor/Counter

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
31	NORMAL	FLOW_UNDERFLOW: FIFO depth is less than threshold - 1 0 = NORMAL 1 = UNDER
30	NORMAL	FLOW_OVERFLOW: FIFO depth is greater than threshold + 1 0 = NORMAL 1 = OVER
4	DISABLE	MONITOR_INT_EN: enable monitor interrupt to APBIF 0 = DISABLE 1 = ENABLE
3	DISABLE	COUNTER_CLR: clear counter 0 = DISABLE 1 = ENABLE
2	DISABLE	MONITOR_CLR: clear monitor 0 = DISABLE 1 = ENABLE
1	DISABLE	COUNTER_EN: enable counter 0 = DISABLE 1 = ENABLE
0	DISABLE	MONITOR_EN: enable monitor 0 = DISABLE 1 = ENABLE

23.3.4.22 SPDIF_FLOW_TOTAL_0

Offset: 0x7c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the total number of left channel samples processed

23.3.4.23 SPDIF_FLOW_OVER_0

Offset: 0x80 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of +STEP_SIZEs

23.3.4.24 SPDIF_FLOW_UNDER_0

Offset: 0x84 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of -STEP_SIZEs

23.3.4.25 SPDIF_INT_STATUS_0

The LCOEF_1_4 coefficients are {46, -1562, 8394, 28999, -3714, 480};

Offset: 0xac | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	USER_TX_DONE: 0 = FALSE 1 = TRUE
14	X	USER_RX_DONE: 0 = FALSE 1 = TRUE
13	X	DATA_TX_DONE: 0 = FALSE 1 = TRUE
12	X	DATA_RX_DONE: 0 = FALSE 1 = TRUE
11	X	USER_TXCIF_OVERRUN: 0 = FALSE 1 = TRUE
10	X	DATA_TXCIF_OVERRUN: 0 = FALSE 1 = TRUE
9	X	USER_RXCIF_UNDERRUN: 0 = FALSE 1 = TRUE
8	X	DATA_RXCIF_UNDERRUN: 0 = FALSE 1 = TRUE
4	X	RX_IU: 0 = FALSE 1 = TRUE
3	X	RX_CHANNEL: 0 = FALSE 1 = TRUE
2	X	FLOW_CTL_INT: 0 = FALSE 1 = TRUE

Bit	Reset	Description
1	X	BAD_PREAMBLE: 0 = FALSE 1 = TRUE
0	X	BSYNC: 0 = FALSE 1 = TRUE

23.3.4.26 SPDIF_INT_MASK_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x0000ff1f (0bxxxxxxxxxxxxxxxx11111111xxx11111)

Bit	Reset	Description
15	MASK	USER_TX_DONE: 0 = UNMASK 1 = MASK
14	MASK	USER_RX_DONE: 0 = UNMASK 1 = MASK
13	MASK	DATA_TX_DONE: 0 = UNMASK 1 = MASK
12	MASK	DATA_RX_DONE: 0 = UNMASK 1 = MASK
11	MASK	USER_TXCIF_OVERRUN: 0 = UNMASK 1 = MASK
10	MASK	DATA_TXCIF_OVERRUN: 0 = UNMASK 1 = MASK
9	MASK	USER_RXCIF_UNDERRUN: 0 = UNMASK 1 = MASK
8	MASK	DATA_RXCIF_UNDERRUN: 0 = UNMASK 1 = MASK
4	MASK	RX_IU: 0 = UNMASK 1 = MASK
3	MASK	RX_CHANNEL: 0 = UNMASK 1 = MASK
2	MASK	FLOW_CTL_INT: 0 = UNMASK 1 = MASK
1	MASK	BAD_PREAMBLE: 0 = UNMASK 1 = MASK
0	MASK	BSYNC: 0 = UNMASK 1 = MASK

23.3.4.27 SPDIF_INT_SET_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000xxx00000)

Bit	Reset	Description
15	CLEAR	USER_TX_DONE: 0 = CLEAR 1 = SET

Bit	Reset	Description
14	CLEAR	USER_RX_DONE: 0 = CLEAR 1 = SET
13	CLEAR	DATA_TX_DONE: 0 = CLEAR 1 = SET
12	CLEAR	DATA_RX_DONE: 0 = CLEAR 1 = SET
11	CLEAR	USER_TXCIF_OVERRUN: 0 = CLEAR 1 = SET
10	CLEAR	DATA_TXCIF_OVERRUN: 0 = CLEAR 1 = SET
9	CLEAR	USER_RXCIF_UNDERRUN: 0 = CLEAR 1 = SET
8	CLEAR	DATA_RXCIF_UNDERRUN: 0 = CLEAR 1 = SET
4	CLEAR	RX_IU: 0 = CLEAR 1 = SET
3	CLEAR	RX_CHANNEL: 0 = CLEAR 1 = SET
2	CLEAR	FLOW_CTL_INT: 0 = CLEAR 1 = SET
1	CLEAR	BAD_PREAMBLE: 0 = CLEAR 1 = SET
0	CLEAR	BSYNC: 0 = CLEAR 1 = SET

23.3.4.28 SPDIF_INT_CLEAR_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000xxx00000)

Bit	Reset	Description
15	CLEAR	USER_TX_DONE: 0 = CLEAR 1 = SET
14	CLEAR	USER_RX_DONE: 0 = CLEAR 1 = SET
13	CLEAR	DATA_TX_DONE: 0 = CLEAR 1 = SET
12	CLEAR	DATA_RX_DONE: 0 = CLEAR 1 = SET
11	CLEAR	USER_TXCIF_OVERRUN: 0 = CLEAR 1 = SET
10	CLEAR	DATA_TXCIF_OVERRUN: 0 = CLEAR 1 = SET

Bit	Reset	Description
9	CLEAR	USER_RXCIF_UNDERRUN: 0 = CLEAR 1 = SET
8	CLEAR	DATA_RXCIF_UNDERRUN: 0 = CLEAR 1 = SET
4	CLEAR	RX_IU: 0 = CLEAR 1 = SET
3	CLEAR	RX_CHANNEL: 0 = CLEAR 1 = SET
2	CLEAR	FLOW_CTL_INT: 0 = CLEAR 1 = SET
1	CLEAR	BAD_PREAMBLE: 0 = CLEAR 1 = SET
0	CLEAR	BSYNC: 0 = CLEAR 1 = SET

23.3.4.29 SPDIF_LIVE_STATUS_0

Offset: 0xbc | Read/Write: RO | Reset: 0x0X0X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	TXC_BSY
19	X	USER_TX_FIFO_FULL: 0 = FALSE 1 = TRUE
18	X	USER_TX_FIFO_EMPTY: 0 = FALSE 1 = TRUE
17	X	DATA_TX_FIFO_FULL: 0 = FALSE 1 = TRUE
16	X	DATA_TX_FIFO_EMPTY: 0 = FALSE 1 = TRUE
11	X	USER_RX_FIFO_FULL: 0 = FALSE 1 = TRUE
10	X	USER_RX_FIFO_EMPTY: 0 = FALSE 1 = TRUE
9	X	DATA_RX_FIFO_FULL: 0 = FALSE 1 = TRUE
8	X	DATA_RX_FIFO_EMPTY: 0 = FALSE 1 = TRUE
3	X	USER_TX_ENABLED: 0 = FALSE 1 = TRUE
2	X	USER_RX_ENABLED: 0 = FALSE 1 = TRUE
1	X	DATA_TX_ENABLED: 0 = FALSE 1 = TRUE

Bit	Reset	Description
0	X	DATA_RX_ENABLED: 0 = FALSE 1 = TRUE

23.3.5 AMX Registers

23.3.5.1 AMX_AXBAR_RX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	ACIF_FIFO4_FULL: 0 = FALSE 1 = TRUE
6	X	ACIF_FIFO4_EMPTY: 0 = FALSE 1 = TRUE
5	X	ACIF_FIFO3_FULL: 0 = FALSE 1 = TRUE
4	X	ACIF_FIFO3_EMPTY: 0 = FALSE 1 = TRUE
3	X	ACIF_FIFO2_FULL: 0 = FALSE 1 = TRUE
2	X	ACIF_FIFO2_EMPTY: 0 = FALSE 1 = TRUE
1	X	ACIF_FIFO1_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO1_EMPTY: 0 = FALSE 1 = TRUE

23.3.5.2 AMX_AXBAR_RX_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	RX4_DONE: 0 = CLEAR 1 = SET
2	X	RX3_DONE: 0 = CLEAR 1 = SET
1	X	RX2_DONE: 0 = CLEAR 1 = SET
0	X	RX1_DONE: 0 = CLEAR 1 = SET

23.3.5.3 AMX_AXBAR_RX_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	MASK	RX4_DONE: 0 = UNMASK 1 = MASK
2	MASK	RX3_DONE: 0 = UNMASK 1 = MASK
1	MASK	RX2_DONE: 0 = UNMASK 1 = MASK
0	MASK	RX1_DONE: 0 = UNMASK 1 = MASK

23.3.5.4 AMX_AXBAR_RX_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	CLEAR	RX4_DONE: This bit is auto cleared after an interrupt in generated 0 = CLEAR 1 = SET
2	CLEAR	RX3_DONE: 0 = CLEAR 1 = SET
1	CLEAR	RX2_DONE: 0 = CLEAR 1 = SET
0	CLEAR	RX1_DONE: 0 = CLEAR 1 = SET

23.3.5.5 AMX_AXBAR_RX_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	CLEAR	RX4_DONE: 0 = CLEAR 1 = SET
2	CLEAR	RX3_DONE: 0 = CLEAR 1 = SET
1	CLEAR	RX2_DONE: 0 = CLEAR 1 = SET
0	CLEAR	RX1_DONE: 0 = CLEAR 1 = SET

23.3.5.6 AMX_AXBAR_RX*n*_CIF_CTRL_0

There are four AMX_AXBAR_RX_CIF_CTRL registers, one per RX port (*n* = 1 through 4).

Offset: 0x20 + ([n * 0x4] – 0x4) | Read/Write: R/W | Reset: 0x000X7X00 (0bxx0000000000xxxxx111xxxx0000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RO	X	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RO	X	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.5.7 AMX_AXBAR_TX_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.5.8 AMX_AXBAR_TX_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = CLEAR 1 = SET

23.3.5.9 AMX_AXBAR_TX_INT_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.5.10 AMX_AXBAR_TX_INT_SET_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: This bit is auto cleared after an interrupt is generated 0 = CLEAR 1 = SET

23.3.5.11 AMX_AXBAR_TX_INT_CLEAR_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: 0 = CLEAR 1 = SET

23.3.5.12 AMX_AXBAR_TX_CIF_CTRL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x000X7X00 (0bxx0000000000xxxxx111xxxx0000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RO	X	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x7	AXBAR_BITS: 0 = RSVD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RO	0x1X	CLIENT_BITS: 0 = RSVD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RO	X	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.5.13 AMX_ENABLE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.5.14 AMX_SOFT_RESET_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.5.15 AMX_CG_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable 0 = FALSE 1 = TRUE

23.3.5.16 AMX_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0xX0000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CNFG_ERR: 0 = FALSE 1 = TRUE
8	X	CLKEN: 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: 0 = FALSE 1 = TRUE

23.3.5.17 AMX_INT_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	TX_DONE: 0 = CLEAR 1 = SET
3	X	RX4_DONE: 0 = CLEAR 1 = SET
2	X	RX3_DONE: 0 = CLEAR 1 = SET

Bit	Reset	Description
1	X	RX2_DONE: 0 = CLEAR 1 = SET
0	X	RX1_DONE: 0 = CLEAR 1 = SET

23.3.5.18 AMX_CTRL_0

AMX Control Register

This register specifies the input stream parameters (the number of audio channels in the input stream and number of bits per sample).

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000xxxx0000)

Bit	Reset	Description
15:14	0x0	MSTR_RX_NUM: 0 = RX1 1 = RX2 2 = RX3 3 = RX4
13:12	0x0	RX_DEP: 0 = WT_ON_ALL 1 = WT_ON_ANY 2 = RSVD
11	0x0	RX4_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
10	0x0	RX3_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
9	0x0	RX2_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
8	0x0	RX1_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
3	0x0	RX4_EN: 0 = DISABLE 1 = ENABLE
2	0x0	RX3_EN: 0 = DISABLE 1 = ENABLE
1	0x0	RX2_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RX1_EN: 0 = DISABLE 1 = ENABLE

23.3.5.19 AMX_OUT_BYTE_EN0_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 0 to 31

23.3.5.20 AMX_OUT_BYTE_EN1_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 32 to 63

23.3.6 ADX Registers

23.3.6.1 ADX_AXBAR_RX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.6.2 ADX_AXBAR_RX_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	RX_DONE: 0 = CLEAR 1 = SET

23.3.6.3 ADX_AXBAR_RX_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	RX_DONE: 0 = UNMASK 1 = MASK

23.3.6.4 ADX_AXBAR_RX_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	RX_DONE: This bit is auto cleared after an interrupt is generated 0 = CLEAR 1 = SET

23.3.6.5 ADX_AXBAR_RX_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	RX_DONE: 0 = CLEAR 1 = SET

23.3.6.6 ADX_AXBAR_RX_CIF_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x000X7X00 (0bxx0000000000xxxxx111xxxx0000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RO	X	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RO	X	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE

Bit	R/W	Reset	Description
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.6.7 ADX_AXBAR_TX_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	ACIF_FIFO4_FULL: 0 = FALSE 1 = TRUE
6	X	ACIF_FIFO4_EMPTY: 0 = FALSE 1 = TRUE
5	X	ACIF_FIFO3_FULL: 0 = FALSE 1 = TRUE
4	X	ACIF_FIFO3_EMPTY: 0 = FALSE 1 = TRUE
3	X	ACIF_FIFO2_FULL: 0 = FALSE 1 = TRUE
2	X	ACIF_FIFO2_EMPTY: 0 = FALSE 1 = TRUE
1	X	ACIF_FIFO1_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO1_EMPTY: 0 = FALSE 1 = TRUE

23.3.6.8 ADX_AXBAR_TX_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	TX4_DONE: 0 = CLEAR 1 = SET
2	X	TX3_DONE: 0 = CLEAR 1 = SET
1	X	TX2_DONE: 0 = CLEAR 1 = SET
0	X	TX1_DONE: 0 = CLEAR 1 = SET

23.3.6.9 ADX_AXBAR_TX_INT_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	MASK	TX4_DONE: 0 = UNMASK 1 = MASK
2	MASK	TX3_DONE: 0 = UNMASK 1 = MASK
1	MASK	TX2_DONE: 0 = UNMASK 1 = MASK
0	MASK	TX1_DONE: 0 = UNMASK 1 = MASK

23.3.6.10 ADX_AXBAR_TX_INT_SET_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	CLEAR	TX4_DONE: This bit is auto cleared after an interrupt is generated 0 = CLEAR 1 = SET
2	CLEAR	TX3_DONE: 0 = CLEAR 1 = SET
1	CLEAR	TX2_DONE: 0 = CLEAR 1 = SET
0	CLEAR	TX1_DONE: 0 = CLEAR 1 = SET

23.3.6.11 ADX_AXBAR_TX_INT_CLEAR_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	CLEAR	TX4_DONE: 0 = CLEAR 1 = SET
2	CLEAR	TX3_DONE: 0 = CLEAR 1 = SET
1	CLEAR	TX2_DONE: 0 = CLEAR 1 = SET
0	CLEAR	TX1_DONE: 0 = CLEAR 1 = SET

23.3.6.12 ADX_AXBAR_TX*n*_CIF_CTRL_0

There are 4 ADX AXBAR TX CIF Registers, where n = 1 through 4).

Offset: 0x60 + ([n * 0x04] – 0x04) | Read/Write: R/W | Reset: 0x000X7X00 (0bxx0000000000xxxxx111xxxx0000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RO	X	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RO	X	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.6.13 ADX_ENABLE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.6.14 ADX_SOFT_RESET_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.6.15 ADX_CG_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable for the first 2 channels and global/common logic. 0 = FALSE 1 = TRUE

23.3.6.16 ADX_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0xX0000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CNFG_ERR: 0 = FALSE 1 = TRUE
8	X	CLKEN: 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: 0 = FALSE 1 = TRUE

23.3.6.17 ADX_INT_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	RX_DONE: 0 = CLEAR 1 = SET
3	X	TX4_DONE: 0 = CLEAR 1 = SET
2	X	TX3_DONE: 0 = CLEAR 1 = SET

Bit	Reset	Description
1	X	TX2_DONE: 0 = CLEAR 1 = SET
0	X	TX1_DONE: 0 = CLEAR 1 = SET

23.3.6.18 ADX_CTRL_0

ADX Control Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
11	0x0	TX4_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
10	0x0	TX3_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
9	0x0	TX2_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
8	0x0	TX1_FORCE_DISABLE: 0 = DISABLE 1 = ENABLE
3	0x0	TX4_EN: 0 = DISABLE 1 = ENABLE
2	0x0	TX3_EN: 0 = DISABLE 1 = ENABLE
1	0x0	TX2_EN: 0 = DISABLE 1 = ENABLE
0	0x0	TX1_EN: 0 = DISABLE 1 = ENABLE

23.3.6.19 ADX_IN_BYTE_EN0_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 0 to 31

23.3.6.20 ADX_IN_BYTE_EN1_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 32 to 63

23.3.7 Audio Bridge Registers

23.3.7.1 ABRIDGE_IDLE_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	ENABLE	IDLE_EN: 0 = DISABLE 1 = ENABLE

23.3.7.2 ABRIDGE_STATS_READ_0

This is an array of 2 identical register entries; the register field below applies to each entry.

Offset: 0x4..0xb | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT

23.3.7.3 ABRIDGE_STATS_WRITE_0

This is an array of 2 identical register entries; the register field below applies to each entry.

Offset: 0xc..0x13 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT

23.3.7.4 ABRIDGE_STATS_CLEAR_0

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x14..0x1b | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CLEAR_WRITE_COUNT
0	0x0	CLEAR_READ_COUNT

23.3.7.5 ABRIDGE_SPARE_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SPARE

23.3.7.6 ABRIDGE_APE_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

> ***Note:*** *The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility*

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation, where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	APE_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	APE_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	APE_CCLK_OVERRIDE
17	0x0	APE_RCLK_OVERRIDE
16	0x0	APE_WCLK_OVERRIDE
3	DISABLE	APE_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	APE_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	APE_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	APE_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

23.3.8 OPE Registers

23.3.8.1 OPE_AXBAR_RX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.8.2 OPE_AXBAR_RX_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	RX_DONE: 0 = CLEAR 1 = SET

23.3.8.3 OPE_AXBAR_RX_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	RX_DONE: 0 = UNMASK 1 = MASK

23.3.8.4 OPE_AXBAR_RX_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	RX_DONE: This bit is auto cleared after an interrupt is generated 0 = CLEAR 1 = SET

23.3.8.5 OPE_AXBAR_RX_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	RX_DONE: 0 = CLEAR 1 = SET

23.3.8.6 OPE_AXBAR_RX_CIF_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x000X7700 (0bxx0000000000xxxxx111x1110000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
19:16	RO	X	CLIENT_CHANNELS: Read only = RXCIF_XBAR_CHANNELS; Hardware is tied to what is programmed for the RXCIF XBAR Channels 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
10:8	RW	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.8.7 OPE_AXBAR_TX_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.8.8 OPE_AXBAR_TX_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = CLEAR 1 = SET

23.3.8.9 OPE_AXBAR_TX_INT_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.8.10 OPE_AXBAR_TX_INT_SET_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: This bit is auto cleared after an interrupt is generated 0 = CLEAR 1 = SET

23.3.8.11 OPE_AXBAR_TX_INT_CLEAR_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: 0 = CLEAR 1 = SET

23.3.8.12 OPE_AXBAR_TX_CIF_CTRL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00XX7700 (0bxx000000xxxxxxxx111x1110000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RO	X	AXBAR_CHANNELS: Read only = RXCIF_XBAR_CHANNELS; Hardware is tied to what is programmed for the RXCIF XBAR Channels 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
19:16	RO	X	CLIENT_CHANNELS: Read only = RXCIF_XBAR_CHANNELS; Hardware is tied to what is programmed for the RXCIF XBAR Channels 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD

Bit	R/W	Reset	Description
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.8.13 OPE_ENABLE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.8.14 OPE_SOFT_RESET_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.8.15 OPE_CG_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable for first 2 channels and global/common logic. 0 = FALSE 1 = TRUE

23.3.8.16 OPE_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0xX0000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	SLCG_CLKEN: 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: 0 = FALSE 1 = TRUE

23.3.8.17 OPE_INT_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	TX_DONE: 0 = CLEAR 1 = SET
0	X	RX_DONE: 0 = CLEAR 1 = SET

23.3.8.18 OPE_DIRECTION_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	DIR: 0 = Data flows from MBDRC to PEQ - default 1= Data flows from PEQ to MBDRC

23.3.9 PEQ Registers

23.3.9.1 PEQ_SOFT_RST_0

Note: *The PEQ soft reset cannot be stand-alone (it will affect the whole data flow of the OPE). Thus instead of using this register, software should use the OPE soft reset register to affect both MBDRC and PEQ as well as CIFs.*

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	SOFT_RESET: This bit is Auto Cleared. Resets PEQ logic - FSM and control states 0 = DISABLE: Software should wait until this bit is cleared 1 = ENABLE

23.3.9.2 PEQ_CG_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SLCG_EN: PEQ Idle Detection Enable - Enables Second Level Clock Gating 0 = FALSE 1 = TRUE

23.3.9.3 PEQ_STATUS_0

Offset: 0x8 | Read/Write: RO | Reset: 0xX000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SLCG_CLKEN: Idle Detect Enable 0 = FALSE 1 = TRUE

23.3.9.4 PEQ_CONFIG_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000013 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx010011)

Bit	Reset	Description
5:2	0x4	BIQUAD_STAGES: Number of BiQuad Stages in PEQ Chain (maximum of 12) This value is N-1; for example, to set the stages to 12 (max), write 'd11 to this field.
1	UNBIAS	BIAS_UNBIAS: Rounding Option across all Rounding Operations of MBDRC Biased = Round to plus infinity (both positive and negative numbers) Unbias = Round to plus infinity (positive number) and minus infinity (negative number) 1 = UNBIAS: 0 = BIAS
0	ACTIVE	MODE: Select Mode. When 0 – Bypass When 1 - Active - will do PreGain, BiQuad, PostGain 0 = BYPASS: 1 = ACTIVE

23.3.9.5 PEQ_AHUBRAMCTL_PEQ_CTRL_0

Coefficient RAM, which has BiQuad Coefficients and Pre/Post Gain Scale Factors

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x10..0x13 | Read/Write: R/W | Reset: 0xX0004000 (0bxxxxxxxx00000000x100xxx000000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
23:16	RW	0x0	SEQ_READ_COUNT
14	RW	WRITE	RW: 0 = READ 1 = WRITE
13	RW	DISABLE	ADDR_INIT_EN: 0 = DISABLE 1 = ENABLE
12	RW	DISABLE	SEQ_ACCESS_EN: 0 = DISABLE 1 = ENABLE
8:0	RW	0x0	RAM_ADDR

23.3.9.6 PEQ_AHUBRAMCTL_PEQ_DATA_0

This is an array of 1 identical register entries; the register field below applies to each entry.

Offset: 0x14..0x17 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.3.9.7 PEQ_AHUBRAMCTL_SHIFT_CTRL_0

Shift RAM, which has BiQuad Stage Shift and Pre/Post Shift Factors

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x18..0x1b | Read/Write: R/W | Reset: 0xX0004000 (0bxxxxxxxx00000000x100xxx000000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY

Bit	R/W	Reset	Description
23:16	RW	0x0	SEQ_READ_COUNT
14	RW	WRITE	RW: 0 = READ 1 = WRITE
13	RW	DISABLE	ADDR_INIT_EN: 0 = DISABLE 1 = ENABLE
12	RW	DISABLE	SEQ_ACCESS_EN: 0 = DISABLE 1 = ENABLE
8:0	RW	0x0	RAM_ADDR

23.3.9.8 PEQ_AHUBRAMCTL_SHIFT_DATA_0

This is an array of 1 identical register entries; the register field below applies to each entry.

Offset: 0x1c..0x1f | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.3.10 DMIC Registers

23.3.10.1 DMIC_AXBAR_TX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.10.2 DMIC_AXBAR_TX_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = CLEAR 1 = SET

23.3.10.3 DMIC_AXBAR_TX_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.10.4 DMIC_AXBAR_TX_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: When the software sets these fields, the corresponding interrupts are generated.- 0 = CLEAR 1 = SET

23.3.10.5 DMIC_AXBAR_TX_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: 0 = CLEAR 1 = SET

23.3.10.6 DMIC_AXBAR_TX_CIF_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00007700 (0bxx00000000000000x111x1110000x000)

Bit	Reset	Description
29:24	0x0	FIFO_THRESHOLD
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	Reset	Description
0	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.10.7 DMIC_ENABLE_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.10.8 DMIC_SOFT_RESET_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.10.9 DMIC_CG_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable for first 2 channels and global/common logic. 0 = FALSE 1 = TRUE

23.3.10.10 DMIC_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0xX0000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CNFG_ERR: 0 = FALSE 1 = TRUE
8	X	SLCG_CLKEN: 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: 0 = FALSE 1 = TRUE

23.3.10.11 DMIC_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = CLEAR 1 = SET

23.3.10.12 DMIC_CTRL_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000301 (0bxxxxxxxxxxxxxxxx00000xx11xxx0xx01)

Bit	Reset	Description
16:12	0x0	TRIMMER_SEL
9:8	STEREO	CHANNEL_SELECT: 0 = NONE 1 = LEFT 2 = RIGHT 3 = STEREO
4	LEFT	LRSEL_POLARITY: 0 = LEFT 1 = RIGHT
1:0	OSR128	OSR: 0 = OSR64 1 = OSR128 2 = OSR256

23.3.11 AHC Registers

23.3.11.1 AHC_SOFT_RESET_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.11.2 AHC_CG_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000101 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx1)

Bit	Reset	Description
8	TRUE	SLAVE_SLCG_EN: Second level clock gating enable, 0 = FALSE 1 = TRUE
0	TRUE	MASTER_SLCG_EN: Second level clock gating enable, 0 = FALSE 1 = TRUE

23.3.11.3 AHC_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
27:16	X	REQUEST_TIMEOUT_COUNTER: config access timeout value
13:12	X	LIVE_STATUS: 0 = FALSE 1 = TRUE
9	X	MASTER_CLKEN: 0 = FALSE 1 = TRUE
8	X	SLAVE_CLKEN: 0 = FALSE 1 = TRUE
5	X	HRD_FIFO_FULL: 0 = FALSE 1 = TRUE

Bit	Reset	Description
4	X	HRD_FIFO_EMPTY: 0 = FALSE 1 = TRUE
1	X	HWR_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	HWR_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.11.4 AHC_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CONFIG_REQUEST_TIMEOUT: time out status indication bit 0 = CLEAR 1 = SET

23.3.11.5 AHC_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	CONFIG_REQUEST_TIMEOUT: 0 = UNMASK 1 = MASK

23.3.11.6 AHC_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	CONFIG_REQUEST_TIMEOUT: 0 = FALSE 1 = TRUE

23.3.11.7 AHC_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	CONFIG_REQUEST_TIMEOUT: 0 = FALSE 1 = TRUE

23.3.11.8 AHC_CTRL_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxx000000000100)

Bit	Reset	Description
11:0	0x4	REQUEST_TIMEOUT_COUNT: timeout value in terms of AHUB clock cycles

23.3.11.9 AHC_AHUB_INTR_STATUS_0_0

Offset: 0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	AFC6: 0 = CLEAR 1 = SET
28	X	AFC5: 0 = CLEAR 1 = SET
27	X	AFC4: 0 = CLEAR 1 = SET
26	X	AFC3: 0 = CLEAR 1 = SET
25	X	AFC2: 0 = CLEAR 1 = SET
24	X	AFC1: 0 = CLEAR 1 = SET
22	X	DMIC3: 0 = CLEAR 1 = SET
21	X	DMIC2: 0 = CLEAR 1 = SET
20	X	DMIC1: 0 = CLEAR 1 = SET
17	X	ADX2: 0 = CLEAR 1 = SET
16	X	ADX1: 0 = CLEAR 1 = SET
13	X	AMX2: 0 = CLEAR 1 = SET
12	X	AMX1: 0 = CLEAR 1 = SET
11	X	SFC4: 0 = CLEAR 1 = SET
10	X	SFC3: 0 = CLEAR 1 = SET
9	X	SFC2: 0 = CLEAR 1 = SET
8	X	SFC1: 0 = CLEAR 1 = SET
4	X	I2S5: 0 = CLEAR 1 = SET
3	X	I2S4: 0 = CLEAR 1 = SET

Bit	Reset	Description
2	X	I2S3: 0 = CLEAR 1 = SET
1	X	I2S2: 0 = CLEAR 1 = SET
0	X	I2S1: 0 = CLEAR 1 = SET

23.3.11.10 AHC_AHUB_INTR_STATUS_1_0

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AHC: 0 = CLEAR 1 = SET
28	X	XBAR: 0 = CLEAR 1 = SET
24	X	ADMAIF: 0 = CLEAR 1 = SET
20	X	MIXER1: 0 = CLEAR 1 = SET
16	X	SPDIF1: 0 = CLEAR 1 = SET
15	X	IQC2: Unused, reserved 0 = CLEAR 1 = SET
14	X	IQC1: Unused, reserved 0 = CLEAR 1 = SET
13	X	MDMIF1: 0 = CLEAR 1 = SET
12	X	MDMIF0: 0 = CLEAR 1 = SET
9	X	MVC2: 0 = CLEAR 1 = SET
8	X	MVC1: 0 = CLEAR 1 = SET
4	X	SPKPROT1: 0 = CLEAR 1 = SET
1	X	OPE2: 0 = CLEAR 1 = SET
0	X	OPE1: 0 = CLEAR 1 = SET

23.3.11.11 AHC_AHUB_ENABLE_STATUS_0_0

Offset: 0x30 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	AFC6: 0 = CLEAR 1 = SET
28	X	AFC5: 0 = CLEAR 1 = SET
27	X	AFC4: 0 = CLEAR 1 = SET
26	X	AFC3: 0 = CLEAR 1 = SET
25	X	AFC2: 0 = CLEAR 1 = SET
24	X	AFC1: 0 = CLEAR 1 = SET
22	X	DMIC3: 0 = CLEAR 1 = SET
21	X	DMIC2: 0 = CLEAR 1 = SET
20	X	DMIC1: 0 = CLEAR 1 = SET
17	X	ADX2: 0 = CLEAR 1 = SET
16	X	ADX1: 0 = CLEAR 1 = SET
13	X	AMX2: 0 = CLEAR 1 = SET
12	X	AMX1: 0 = CLEAR 1 = SET
11	X	SFC4: 0 = CLEAR 1 = SET
10	X	SFC3: 0 = CLEAR 1 = SET
9	X	SFC2: 0 = CLEAR 1 = SET
8	X	SFC1: 0 = CLEAR 1 = SET
4	X	I2S5: 0 = CLEAR 1 = SET
3	X	I2S4: 0 = CLEAR 1 = SET

Bit	Reset	Description
2	X	I2S3: 0 = CLEAR 1 = SET
1	X	I2S2: 0 = CLEAR 1 = SET
0	X	I2S1: 0 = CLEAR 1 = SET

23.3.11.12 AHC_AHUB_ENABLE_STATUS_1_0

Offset: 0x34 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AHC: 0 = CLEAR 1 = SET
28	X	XBAR: 0 = CLEAR 1 = SET
24	X	ADMAIF: 0 = CLEAR 1 = SET
20	X	MIXER1: 0 = CLEAR 1 = SET
16	X	SPDIF1: 0 = CLEAR 1 = SET
15	X	IQC2: Unused, reserved 0 = CLEAR 1 = SET
14	X	IQC1: Unused, reserved 0 = CLEAR 1 = SET
13	X	MDMIF1: 0 = CLEAR 1 = SET
12	X	MDMIF0: 0 = CLEAR 1 = SET
9	X	MVC2: 0 = CLEAR 1 = SET
8	X	MVC1: 0 = CLEAR 1 = SET
4	X	SPKPROT1: 0 = CLEAR 1 = SET
1	X	OPE2: 0 = CLEAR 1 = SET
0	X	OPE1: 0 = CLEAR 1 = SET

23.3.12 AMC Registers

23.3.12.1 AMC_CONFIG_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	FALSE	ERR_RESPONSE_DISABLE: 0 = FALSE 1 = TRUE
1	FALSE	CARVE_OUT_ENABLE: 0 = FALSE 1 = TRUE
0	FALSE	ARAM_ADDR_ALIAS_ENABLE: 0 = FALSE 1 = TRUE

23.3.12.2 AMC_INT_STATUS_0

Offset: 0x4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	INVALID_CONFIG_ADDR: 0 = FALSE 1 = TRUE
2	X	INVALID_BURST_TYPE: 0 = FALSE 1 = TRUE
1	X	INVALID_REQUEST_TYPE: 0 = FALSE 1 = TRUE
0	X	INVALID_AMEM_ACCESS: 0 = FALSE 1 = TRUE

23.3.12.3 AMC_INT_MASK_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	MASK	INVALID_CONFIG_ADDR: 0 = UNMASK 1 = MASK
2	MASK	INVALID_BURST_TYPE: 0 = UNMASK 1 = MASK
1	MASK	INVALID_REQUEST_TYPE: 0 = UNMASK 1 = MASK
0	MASK	INVALID_AMEM_ACCESS: 0 = UNMASK 1 = MASK

23.3.12.4 AMC_INT_SET_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	FALSE	INVALID_CONFIG_ADDR: 0 = FALSE 1 = TRUE

Bit	Reset	Description
2	FALSE	INVALID_BURST_TYPE: 0 = FALSE 1 = TRUE
1	FALSE	INVALID_REQUEST_TYPE: 0 = FALSE 1 = TRUE
0	FALSE	INVALID_AMEM_ACCESS: 0 = FALSE 1 = TRUE

23.3.12.5 AMC_INT_CLEAR_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	FALSE	INVALID_CONFIG_ADDR: 0 = FALSE 1 = TRUE
2	FALSE	INVALID_BURST_TYPE: 0 = FALSE 1 = TRUE
1	FALSE	INVALID_REQUEST_TYPE: 0 = FALSE 1 = TRUE
0	FALSE	INVALID_AMEM_ACCESS: 0 = FALSE 1 = TRUE

23.3.12.6 AMC_ERROR_ADDR_0

Offset: 0x14 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ERROR_ADDR

23.3.12.7 AMC_APERTURE_BASE_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0000000000xxxxxxxxxx)

Bit	Reset	Description
20:11	0x0	APERTURE_BASE

23.3.12.8 AMC_EVP_RESET_VEC_0

Offset: 0x700 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RESET_VECTOR

23.3.12.9 AMC_EVP_UNDEF_VEC_0

Offset: 0x704 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	UNDEF_VECTOR

23.3.12.10 AMC_EVP_SWI_VEC_0

Offset: 0x708 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SWI_VECTOR

23.3.12.11 AMC_EVP_PREFETCH_ABORT_VEC_0

Offset: 0x70c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PREFETCH_ABORT_VECTOR

23.3.12.12 AMC_EVP_DATA_ABORT_VEC_0

Offset: 0x710 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA_ABORT_VECTOR

23.3.12.13 AMC_EVP_RSVD_VEC_0

Offset: 0x714 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RSVD_VECTOR

23.3.12.14 AMC_EVP_IRQ_VEC_0

Offset: 0x718 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	IRQ_VECTOR

23.3.12.15 AMC_EVP_FIQ_VEC_0

Offset: 0x71c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	FIQ_VECTOR

23.3.12.16 AMC_EVP_RESET_ADDR_0

Offset: 0x720 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RESET_ADDR

23.3.12.17 AMC_EVP_UNDEF_ADDR_0

Offset: 0x724 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	UNDEF_ADDR

23.3.12.18 AMC_EVP_SWI_ADDR_0

Offset: 0x728 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SWI_ADDR

23.3.12.19 AMC_EVP_PREFETCH_ABORT_ADDR_0

Offset: 0x72c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PREFETCH_ABORT_ADDR

23.3.12.20 AMC_EVP_DATA_ABORT_ADDR_0

Offset: 0x730 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA_ABORT_ADDR

23.3.12.21 AMC_EVP_RSVD_ADDR_0

Offset: 0x734 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RSVD_ADDR

23.3.12.22 AMC_EVP_IRQ_ADDR_0

Offset: 0x738 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	IRQ_ADDR

23.3.12.23 AMC_EVP_FIQ_ADDR_0

Offset: 0x73c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

APERTURE_DATA

Bit	Reset	Description
31:0	X	FIQ_ADDR

23.3.12.24 AMC_APERTURE_DATA_0

This is an array of 512 identical register entries; the register fields below apply to each entry.

Offset: 0x800..0xfff | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

23.3.13 AFC Registers

23.3.13.1 AFC_AXBAR_RX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.13.2 AFC_AXBAR_RX_CIF_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00007700 (0bxx00000000000000x111x1110000x000)

Bit	Reset	Description
29:24	0x0	FIFO_THRESHOLD
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.13.3 AFC_AXBAR_TX_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
1	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	TX_ENABLED: 0 = FALSE 1 = TRUE

23.3.13.4 AFC_AXBAR_TX_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = CLEAR 1 = SET

23.3.13.5 AFC_AXBAR_TX_INT_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.13.6 AFC_AXBAR_TX_INT_SET_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: 0 = CLEAR 1 = SET

23.3.13.7 AFC_AXBAR_TX_INT_CLEAR_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: 0 = CLEAR 1 = SET

23.3.13.8 AFC_AXBAR_TX_CIF_CTRL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x000X7X00 (0bxx0000000000xxxxx111xxxx0000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2

Bit	R/W	Reset	Description
19:16	RO	X	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RO	X	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.13.9 AFC_ENABLE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.13.10 AFC_SOFT_RESET_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.13.11 AFC_CG_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable for first 2 channels and global/common logic. 0 = FALSE 1 = TRUE

23.3.13.12 AFC_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0xX00000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CNFG_ERR: 0 = FALSE 1 = TRUE
5	X	INTERP_RATE_SLOW: 0 = FALSE 1 = TRUE
4	X	DECIM_RATE_SLOW: 0 = FALSE 1 = TRUE
2	X	CLKEN: 0 = FALSE 1 = TRUE
0	X	TX_ENABLED: 0 = FALSE 1 = TRUE

23.3.13.13 AFC_INT_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	TX_DONE: 0 = CLEAR 1 = SET
0	X	CLK_PPM_DIFF_ERROR: 0 = CLEAR 1 = SET

23.3.13.14 AFC_INT_MASK_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	CLK_PPM_DIFF_ERROR: 0 = UNMASK 1 = MASK

23.3.13.15 AFC_INT_SET_0

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	CLK_PPM_DIFF_ERROR: 0 = CLEAR 1 = SET

23.3.13.16 AFC_INT_CLEAR_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	CLK_PPM_DIFF_ERROR: 0 = CLEAR 1 = SET

23.3.13.17 AFC_DEST_I2S_PARAMS_0

DEST_I2S_PARAMS Register

This register contains parameters concerning the destination I2S (AFC is plugged in somewhere in a path between a source I2S/DMIC and a destination I2S).

Offset: 0xa4 | Read/Write: R/W | Reset: 0x01190e0c (0bxxxxx001x0011001x0001110x0001100)

Bit	Reset	Description
26:24	I2S1	I2S_ID: The ID of the destination I2S (number from 1 to N, N being the last I2S ID for a given chip) 1 = I2S1 2 = I2S2 3 = I2S3 4 = I2S4 5 = I2S5
22:16	0x19	FIFO_HIGH_THRESHOLD: The RxCIF FIFO threshold of the destination I2S. If the number of samples in the FIFO goes above this threshold, the AFC starts decimation.
14:8	0xe	FIFO_START_THRESHOLD: The RxCIF FIFO threshold of the destination I2S. If this threshold is breached, the AFC will start its operations. Until then AFC will be a simple bypass.
6:0	0xc	FIFO_LOW_THRESHOLD: The RxCIF FIFO threshold of the destination I2S. If the number of samples in the FIFO goes below this threshold, the AFC starts interpolation.

23.3.13.18 AFC_TXCIF_FIFO_PARAMS_0

TXCIF_FIFO_PARAMS Register

This register contains parameters concerning AFC TxCIF FIFO.

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00190e0c (0bxxxxxxxxxx0011001x0001110x0001100)

Bit	Reset	Description
22:16	0x19	FIFO_HIGH_THRESHOLD: The FIFO threshold of the destination I2S. If the number of samples in the FIFO goes above this threshold, the AFC starts decimation.
14:8	0xe	FIFO_START_THRESHOLD: The FIFO threshold of the destination I2S. If the threshold is breached, the AFC will start its operations. Until then AFC will be a simple bypass.
6:0	0xc	FIFO_LOW_THRESHOLD: The FIFO threshold of the destination I2S. If the number of samples in the FIFO goes below this threshold, the AFC starts interpolation.

23.3.13.19 AFC_CLK_PPM_DIFF_0

CLK_PPM_DIFF

This register is used to specify the maximum PPM difference between the source I2S/DMIC and the destination I2S clocks. For example, if the crystals/PLLs used to generate clocks for the source and the destination both have a specified maximum PPM of +/- 10ppm, then it is advisable to set this register to a value greater than 20.

Offset: 0xac | Read/Write: R/W | Reset: 0x0000001e (0bxxxxxxxxxxxxxxxx0000000000011110)

Bit	Reset	Description
15:0	0x1e	VALUE: Most codecs are expected to have +/- 10 ppm. So the maximum difference is 20 ppm. A value of 30 is chosen (greater than 20).

23.3.14 MVC Registers

23.3.14.1 MVC_AXBAR_RX_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.14.2 MVC_AXBAR_RX_INT_STATUS_0

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	RX_DONE: 0 = CLEAR 1 = SET

23.3.14.3 MVC_AXBAR_RX_INT_MASK_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	RX_DONE: 0 = UNMASK 1 = MASK

23.3.14.4 MVC_AXBAR_RX_INT_SET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	RX_DONE: This bit is auto cleared after an interrupt is generated 0 = FALSE 1 = TRUE

23.3.14.5 MVC_AXBAR_RX_INT_CLEAR_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	RX_DONE: 0 = FALSE 1 = TRUE

23.3.14.6 MVC_AXBAR_RX_CIF_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x000X7700 (0bxx0000000000xxxxx111x1110000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD
23:20	RW	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
19:16	RO	X	CLIENT_CHANNELS: Equal to RX_CIF_CTRL AXBAR_CHANNELS 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.14.7 MVC_AXBAR_TX_STATUS_0

Offset: 0x4c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.14.8 MVC_AXBAR_TX_INT_STATUS_0

Offset: 0x50 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = CLEAR 1 = SET

23.3.14.9 MVC_AXBAR_TX_INT_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.14.10 MVC_AXBAR_TX_INT_SET_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	TX_DONE: This bit is auto cleared after an interrupt is generated 0 = FALSE 1 = TRUE

23.3.14.11 MVC_AXBAR_TX_INT_CLEAR_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	TX_DONE: 0 = FALSE 1 = TRUE

23.3.14.12 MVC_AXBAR_TX_CIF_CTRL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00XX7700 (0bxx000000xxxxxxxx111x1110000x000)

Bit	R/W	Reset	Description
29:24	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
23:20	RO	X	AXBAR_CHANNELS: Equal to RX_CIF_CTRL AXBAR_CHANNELS 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
19:16	RO	X	CLIENT_CHANNELS: Equal to RX_CIF_CTRL AXBAR_CHANNELS 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
14:12	RW	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	RW	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.14.13 MVC_ENABLE_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: When disabled, the MVC is not disabled until all the data in the data pipe has been processed and sent out. Read back of the register also gives the module enable status 0 = FALSE 1 = TRUE

23.3.14.14 MVC_SOFT_RESET_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: Soft Reset is a self-clearing bit. Soft Reset resets all FSMs, flushes flow control FIFO and resets the state registers. It also bring the module back to disabled state (without flushing data in the pipe) 0 = FALSE 1 = TRUE

23.3.14.15 MVC_CG_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable 0 = FALSE 1 = TRUE

23.3.14.16 MVC_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0xX0000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	SLCG_CLKEN: Status of SLCG 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: Status of MVC enabling. 0 = FALSE 1 = TRUE

23.3.14.17 MVC_INT_STATUS_0

Offset: 0x94 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	TX_DONE: When the MVC is disabled, and all pipe data (RXCIF, MVC CORE and TXCIF) is flushed, TX_DONE is issued. 0 = CLEAR 1 = SET
0	X	RX_DONE: When the MVC is disabled, RXCIF stops receiving data from AXBAR. RX_DONE will be issued after all the data in the TXCIF FIFO is popped out. 0 = CLEAR 1 = SET

23.3.14.18 MVC_CTRL_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x40000001 (0b01xxxxxxxxxxxxxx00000000xxxxxx01)

Bit	Reset	Description
15:8	0x0	MUTE_UNMUTE: Mute/UnMute control bit. 8 bits for 8 channels. bit[8]->ch0, bit[9]->ch1... bit[15]->ch7. If PER_CHAN_CTRL_EN is DISABLE, all the channels use the same MUTE_UNMUTE control: ch0->MUTE_UNMUTE[0] If software programs this register, must be followed by VOLUME_SWITCH trigger to let hardware know new volume parameters are ready. Hardware will take the new target volume (mute value or previous unmute target volume) after finishing current frame. If software programs new target volume when system is under MUTE, this bit will be set to UNMUTE by hardware. Usually, this case is not allowed: Software programming both MUTE and new target volume at the same time. But if it is set by accident, the system will go to MUTE. 0 = UNMUTE 255 = MUTE

Bit	Reset	Description
1	0x0	CURVE_TYPE: Curve type. 0 = POLYNOMIAL Curve 1 = LINEAR RAMP curve Curve type switching cannot be done on the fly Software reset or disable/enable is needed to switch between two curves. 0 = POLYNOMIAL 1 = LINEAR_RAMP
0	UNBIASED	ROUNDING_TYPE: Rounding option. 1 = unbiased rounding (RNE) 0 = biased rounding 0 = BIASED 1 = UNBIASED

23.3.14.19 MVC_SWITCH_0

This register contains three switch bits (Volume, Coefficient, and Duration) for when software wants to do switching for all three parameters at the same time. Software just programs the three bits in this register, Hardware takes the new set of volume, duration, and coefficient simultaneously when processing new frame samples. There is interdependency between this and the coefficient switch, which means software cannot issue this switch until both coefficient switch and duration switch are cleared.

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	CLEAR	VOLUME_SWITCH: Software should write to this field to prompt the hardware to start using a new volume set for the volume related parameters (TARGET_VOL AND MUTE_UNMUTE), 1 bit for 8 channels. Hardware will detect which channel has a new target and do volume ramping. After software programs a new TARGET_VOL or changes MUTE_UNMUTE status, software needs to program this field to let hardware know a new volume is ready for using. Hardware does this switch as soon as it is done processing the current frame. This bit will remain 1 (SET) until the switch has happened, and is cleared to 0 by hardware as an ack that the configuration has been switched to the new one. During Write - The field does not matter, only a Write Access to this field to generate a trigger pulse. During Read - The field indicates whether polynomial coefficients switch clearing has happened or not 0 = CLEAR 1 = SET
1	CLEAR	COEFF_SWITCH: Software should write to this field - to prompt the hardware to start using a new configuration set for the polynomial coefficients. After software programs new polynomial coefficients, software needs to program this field to let hardware know new volume is ready for using. Hardware does this switch as soon as it is done processing the current frame. This bit will remain 1(SET) until the switch has happened, and is cleared to 0 by hardware as an ack that the configuration has been switched to the new one. During Write - The field does not matter, only a Write Access to this field to generate a trigger pulse. During Read - The field indicates whether polynomial coefficients switch clearing has happened or not there is interdependent between this and duration switch, which means software cannot issue this switch until both coefficient switch and duration switch are cleared 0 = CLEAR 1 = SET
0	CLEAR	DURATION_SWITCH: Software should write to this field to prompt the hardware to start using a new configuration set for the duration related parameters (DURATION, DURATION_INV, POLY_N1 AND POLY_N2). After software programs new duration parameters, software needs to program this field to let hardware know a new volume is ready for using. Hardware does this switch as soon as it is done processing the current frame. This bit will remain 1 (SET) until the switch has happened, and is cleared to 0 by hardware as an ack that the configuration has been switched to the new one. During Write - The field does not matter, only a Write Access to this field to generate a trigger pulse. During Read - The field indicates whether duration parameters switch clearing has happened or not. 0 = CLEAR 1 = SET

23.3.14.20 MVC_INIT_VOL_0

This is an array of 8 identical register entries; the register fields below apply to each entry.

Offset: 0xb0..0xcf | Read/Write: R/W | Reset: 0x00800000 (0b00000000100000000000000000000000)

Bit	Reset	Description
31:0	0x800000	INIT_VOL: 8 initial volumes for 8 channels. If PER_CHAN_CTRL_EN is DISABLE, all the channels use the same initial volume: ch0->INIT_VOL_0. This register is shared for Volume Initialization for two curve types: Polynomial: default curve type, all 32 bits of the registers are used. The volume is provided in linear and should be in signed Q8.24 format. Default: 0x80_0000 (0.5, Q8.24). Range: 0x0 ~ 0x6400_0000 (0 ~ 100 @Q8.24). Positive value only, cannot be set with negative number. Linear Ramp: For this curve type, only bottom 16 bits are used. The volume is provided in dB and should be in signed Q8.8 format. Default: 0xffff_fa00 (-6dB @Q8.8). Range: 0xffff_8800 ~ 0x2800 (-120dB ~ 40dB @Q8.8)

23.3.14.21 MVC_TARGET_VOL_0

This is an array of 8 identical register entries; the register fields below apply to each entry.

Offset: 0xd0..0xef | Read/Write: R/W | Reset: 0x00800000 (0b00000000100000000000000000000000)

Bit	Reset	Description
31:0	0x800000	TARGET_VOL: 8 target volumes for 8 channels. If PER_CHAN_CTRL_EN is DISABLE, all the channels use the same target volume: ch0->TARGET_VOL_0. This register is shared for Target Volume for two curve types: Polynomial: For this curve type, all 32 bits of the registers are used. The volume is provided in linear and should be in signed 8.24 format. Default: 0x80_0000 (0.5, Q8.24). Range: 0x0 ~ 0x6400_0000 (0 ~ 100 @Q8.24). Positive value only, cannot be set with negative number. Linear Ramp: For this curve type, only bottom 16 bits are used. The volume is provided in dB and should be in signed 8.8 format. Default: 0xffff_fa00 (-6dB @Q8.8). Range: 0xffff_8800 ~ 0x2800 (-120dB ~ 40dB @Q8.8) If software programs this register, must be followed by VOLUME_SWITCH trigger to let hardware know new volume parameters are ready. Hardware will take the new target volume after finishing current frame. Usually this case is not allowed: Software programming both MUTE and new target volume at the same time. But if it is by accident, the system will go to MUTE, ignore the target volume set it with RWTO, which hardware can update this register to a MUTE value when software is programming the mute flag

23.3.14.22 MVC_DURATION_0

Offset: 0xf0 | Read/Write: R/W | Reset: 0x000012c0 (0bxxxxxxxx00000000000001001011000000)

Bit	Reset	Description
23:0	0x12c0	DURATION: Number of samples to reach the target volume default:0x12C0 (100ms @ 48KHz). If software programs this register, it must be followed by a DURATION_SWITCH trigger to let hardware know new duration parameters are ready. Hardware will take the new duration parameter after finishing the current frame.

23.3.14.23 MVC_DURATION_INV_0

Offset: 0xf4 | Read/Write: R/W | Reset: 0x0006d3a0 (0b00000000000001101101001110100000)

Bit	Reset	Description
31:0	0x6d3a0	DURATION_INV: inverse ramp duration in samples. If software programs this register, it must be followed by a DURATION_SWITCH trigger to let hardware know new duration parameters are ready. Hardware will take the new inv_duration parameter after finishing the current frame.

23.3.14.24 MVC_POLY_N1_0

Offset: 0xf8 | Read/Write: R/W | Reset: 0x0000007d (0bxxxxxxxx00000000000000000001111101)

Bit	Reset	Description
23:0	0x7d	POLY_N1: The first splitting points for the three segments POLY_N1, POLY_N2, and DURATION programming should meet POLY_N1 <= POLY_N2 <=DURATION. For two segments case: POLY_N2 == DURATION. Programming coefficients for the first two segments: p1_0/1/2 and p2_0/1/2. For one segment case: POLY_N1 == POLY_N2 == DURATION. Programming coefficients for the first segment: p1_0/1/2. If software programs this register, it must be followed by DURATION_SWITCH trigger to let hardware know new duration parameters are ready. Hardware will take the new poly_n1 parameter after finishing current frame. Note: POLY_N1 cannot be changed without DURATION changing.

23.3.14.25 MVC_POLY_N2_0

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000271 (0bxxxxxxxx00000000000000001001110001)

Bit	Reset	Description
23:0	0x271	POLY_N2: The second splitting points for the three segments POLY_N1, POLY_N2, and DURATION programming should meet POLY_N1 <= POLY_N2 <=DURATION. For two segments case: POLY_N2 == DURATION. Programming coefficients for the first two segments: p1_0/1/2 and p2_0/1/2 For one segment case: POLY_N1 == POLY_N2 == DURATION. Programming coefficients for the first segment: p1_0/1/2 If software programs this register, it must be followed by DURATION_SWITCH trigger to let hardware know new duration parameters are ready. Hardware will take the new poly_n2 parameter after finishing current frame. Note: POLY_N2 cannot be changed without DURATION changing.

23.3.14.26 MVC_PEAK_CTRL_0

Offset: 0x100 | Read/Write: R/W | Reset: 0x000012c0 (0b0xxxxxxx00000000000001001011000000)

Bit	Reset	Description
31	0x0	PEAK_TYPE: Peak metering type: 0 for window based (default), 1 for reset-on-read 0 = WINPEAK 1 = RORPEAK
23:0	0x12c0	PEAK_WIN: It is for Peak Metering window size, as the number of samples. Only available when peak metering type is window based peak metering. Default:0x12C0 (100ms @ 48KHz)

23.3.14.27 MVC_AHUBRAMCTL_CONFIG_RAM_CTRL_0

MVC Configuration RAM

1. The content in the RAM is shown below
2. Only one Configuration RAM

   0x00: poly_coeff_p1_0 // 0x01: poly_coeff_p1_1 |

   0x02: poly_coeff_p1_2 |

   0x03: poly_coeff_p2_0 \ PING

   0x04: poly_coeff_p2_1 /

   0x05: poly_coeff_p2_2 |

   0x06: poly_coeff_p3_0 |

   0x07: poly_coeff_p3_1 |

   0x08: poly_coeff_p3_2 /

0x09: poly_coeff_p1_0 // 0x0a: poly_coeff_p1_1 |

0x0b: poly_coeff_p1_2 |

0x0c: poly_coeff_p2_0 \ PONG

0x0d: poly_coeff_p2_1 /

0x0e: poly_coeff_p2_2 |

0x0f: poly_coeff_p3_0 |

0x10: poly_coeff_p3_1 |

0x11: poly_coeff_p3_2 /

3. PING/PONG scheme, 0x0~0x8 is PING, and 0x9~0x11 is PONG.

   PING-PONG scheme is taken care by hardware, which means software does not need to care which part (Ping or Pong) in the buffer to be programmed.

   Just set the base address to "0", and hardware will handle the PING-PONG scheme.

4. Before software programming, software needs to check COEFF_SWITCH status.

   If it is cleared, then software can start program new polynomial coefficients.

   After software finishes programming, it needs to write COEFF_SWITCH for hardware switching.

   Hardware will know the new configuration parameters are ready in the memory.

5. Poly coefficients usage:

   Ramp up:

   c[n] = p1_0*(n/N)^2 + p1_1*(n/N)^2 + p1_2 (n is in [0,POLY_N1))

   c[n] = p2_0*(n/N)^2 + p2_1*(n/N)^2 + p2_2 (n is in [POLY_N1,POLY_N2))

   c[n] = p3_0*(n/N)^2 + p3_1*(n/N)^2 + p3_2 (n is in [POLY_N2,DURATION))

   Ramp up:

   c[n] = p1_0*((N-n)/N)^2 + p1_1*((N-n)/N)^2 + p1_2 (n is in [0,POLY_N1))

   c[n] = p2_0*((N-n)/N)^2 + p2_1*((N-n)/N)^2 + p2_2 (n is in [POLY_N1,POLY_N2))

   c[n] = p3_0*((N-n)/N)^2 + p3_1*((N-n)/N)^2 + p3_2 (n is in [POLY_N2,DURATION))

6. For two segments case: only programming p1_0/1/2 and p2_0/1/2; POLY_N2==DURATION

7. For one segment case: only programming p1_0/1/2; POLY_N1==POLY_N2==DURATION

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x104..0x107 | Read/Write: R/W | Reset: 0xX0004000 (0bxxxxxxxx00000000x100xxx000000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
23:16	RW	0x0	SEQ_READ_COUNT
14	RW	WRITE	RW: 0 = READ 1 = WRITE
13	RW	DISABLE	ADDR_INIT_EN: 0 = DISABLE 1 = ENABLE
12	RW	DISABLE	SEQ_ACCESS_EN: 0 = DISABLE 1 = ENABLE
8:0	RW	0x0	RAM_ADDR

23.3.14.28 MVC_AHUBRAMCTL_CONFIG_RAM_DATA_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x108..0x10b | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.3.14.29 MVC_PEAK_VALUE_0

This is an array of 8 identical register entries; the register fields below apply to each entry.

Offset: 0x10c..0x12b | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PEAK_VALUE: peak value for software read, 8 for 7.1 audio format

23.3.14.30 MVC_CONFIG_ERR_TYPE_0

Offset: 0x12c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	VOLUME_CONFIG_ERROR: Two sources: 1. Software programming target volume during Switching is set 2. Wrong programming: a. INIT_VOL > max volume, or INIT_VOL < min volume (refer to range of INIT_VOL) b. TARGET_VOL > max volume, or TARGET_VOL < min volume (refer to range of TARGET_VOL) 0 = FALSE 1 = TRUE
1	X	COEFF_CONFIG_ERROR: Software programming coefficient buffer during Switching is set 0 = FALSE 1 = TRUE
0	X	DURATION_CONFIG_ERROR: There are two sources for this error: 1. Software programming duration parameters (DURATION, DURATION_INV, POLY_N1 and POLY_N2) during Switching is SET 2. Wrong programming: a. POLY_N1 > POLY_N2 or POLY_N1 > DURATION or POLY_N2 > DURATION b. BOTH DURATION and DURATION_INV should be changed simultaneously. Only one of them changed will result in error config 0 = FALSE 1 = TRUE

23.3.15 DMA Registers

23.3.15.1 ADMA_CH<*n*>_CMD_0

There are 22 ADMA Command registers, one per channel (n = 1 through 22).

Offset: 0x0 + (n – 1) * 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	TRANSFER_ENABLE: Enables DMA transfer. This bit can be cleared by either software or hardware. Hardware clears this bit when the response/acknowledgment from the target device is received for the last write request of the transfer in once and linked list modes. Clearing this bit by software causes disabling the channel and all transfers initiated prior to it completed normally. Software should set this bit after configuring all other registers of this channel. 0 = FALSE 1 = TRUE

23.3.15.2 ADMA_CH<*n*>_SOFT_RESET_0

There are 22 ADMA Soft Reset registers, one per channel (n = 1 through 22).

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE: Self clearing software initiated reset. 0 = FALSE 1 = TRUE

23.3.15.3 ADMA_CH<*n*>_STATUS_0

There are 22 ADMA Status registers, one per channel (n = 1 through 22).

Offset: 0xc | Read/Write: RO | Reset: 0x00XX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
22:20	X	CURRENT_SOURCE_MEMORY_BUFFER: Indicates source memory buffer that is currently being read by ADMA. 0 = BUFFER_1 1 = BUFFER_2 2 = BUFFER_3 3 = BUFFER_4 4 = BUFFER_5 5 = BUFFER_6 6 = BUFFER_7 7 = BUFFER_8
18:16	X	CURRENT_TARGET_MEMORY_BUFFER: Indicates target memory buffer that is currently being written into by ADMA. 0 = BUFFER_1 1 = BUFFER_2 2 = BUFFER_3 3 = BUFFER_4 4 = BUFFER_5 5 = BUFFER_6 6 = BUFFER_7 7 = BUFFER_8
2	X	OUTSTANDING_TRANSFERS: Indicates if there is any DMA request pending to be completed. 0 = FALSE 1 = TRUE
1	X	TRANSFER_PAUSED: Asserts when "TRANSFER_PAUSE" bit is set by software and all previously issued DMA requests are completed. Software should attempt to access free running status bits like transfer count when this status bit goes high. 0 = FALSE 1 = TRUE
0	X	TRANSFER_ENABLED: This bit is set by hardware when software enabled the channel by setting TRANSFER_ENABLE control bit and is cleared when transfer is completed or when TRANSFER_ENABLE bit is cleared by software and all the previously issued requests are completed. 0 = FALSE 1 = TRUE

23.3.15.4 ADMA_CH<*n*>_INT_STATUS_0

There are 22 ADMA Interrupt Status registers, one per channel (n = 1 through 22).

Offset: 0x10 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TRANSFER_DONE: Pre mask status of hardware or software initiated interrupt. Hardware generates an interrupt when it receives response/commit for last write request issued to the target. 0 = FALSE 1 = TRUE

23.3.15.5 ADMA_CH<*n*>_INT_SET_0

There are 22 ADMA Interrupt Set registers, one per channel (n = 1 through 22).

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	TRANSFER_DONE: Software may choose to set this interrupt for debug/development purposes. 0 = FALSE 1 = TRUE

23.3.15.6 ADMA_CH<*n*>_INT_CLEAR_0

There are 22 ADMA Interrupt Clear registers, one per channel (n = 1 through 22).

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	TRANSFER_DONE: Software should set this bit to clear this interrupt when it is serviced. 0 = FALSE 1 = TRUE

23.3.15.7 ADMA_CH<*n*>_CTRL_0

There are 22 ADMA Control registers, one per channel (n = 1 through 22).

Offset: 0x24 + (n – 1) * 0x80 | Read/Write: R/W | Reset: 0x11004102 (0b00010001xxx000000100x001xxxxx010)

Bit	Reset	Description
31:28	0x1	TX_REQUEST_SELECT: One of the AHUB transmit channels mapped as a target for DMA transfer by enabling "FLOWCTRL_ENABLE" bit. 1 = AHUB_CH1_REQUEST 2 = AHUB_CH2_REQUEST 3 = AHUB_CH3_REQUEST 4 = AHUB_CH4_REQUEST 5 = AHUB_CH5_REQUEST 6 = AHUB_CH6_REQUEST 7 = AHUB_CH7_REQUEST 8 = AHUB_CH8_REQUEST 9 = AHUB_CH9_REQUEST 10 = AHUB_CH10_REQUEST
27:24	0x1	RX_REQUEST_SELECT: One of the AHUB receive channels mapped as a source for DMA transfer by enabling the "FLOWCTRL_ENABLE" bit. 1 = AHUB_CH1_REQUEST 2 = AHUB_CH2_REQUEST 3 = AHUB_CH3_REQUEST 4 = AHUB_CH4_REQUEST 5 = AHUB_CH5_REQUEST 6 = AHUB_CH6_REQUEST 7 = AHUB_CH7_REQUEST 8 = AHUB_CH8_REQUEST 9 = AHUB_CH9_REQUEST 10 = AHUB_CH10_REQUEST

Bit	Reset	Description
20:16	0x0	TRIGGER_SELECT: One of the "TRANSFER_DONE" interrupts of other active channels to trigger the DMA transfer from this channel. This would be valid when the "TRIGGER_ENABLE" bit is set. 1 = CH1_TRANSFER_DONE 2 = CH2_TRANSFER_DONE 3 = CH3_TRANSFER_DONE 4 = CH4_TRANSFER_DONE 5 = CH5_TRANSFER_DONE 6 = CH6_TRANSFER_DONE 7 = CH7_TRANSFER_DONE 8 = CH8_TRANSFER_DONE 9 = CH9_TRANSFER_DONE 10 = CH10_TRANSFER_DONE 11 = CH11_TRANSFER_DONE 12 = CH12_TRANSFER_DONE 13 = CH13_TRANSFER_DONE 14 = CH14_TRANSFER_DONE 15 = CH15_TRANSFER_DONE 16 = CH16_TRANSFER_DONE 17 = CH17_TRANSFER_DONE 18 = CH18_TRANSFER_DONE 19 = CH19_TRANSFER_DONE 20 = CH20_TRANSFER_DONE 21 = CH21_TRANSFER_DONE 22 = CH22_TRANSFER_DONE
15:12	0x4	TRANSFER_DIRECTION: Indicates the source and destination of the DMA transfer. Possible transfer directions are Memory-to-Memory, AHUB-to-AHUB. Memory <--> AHUB. Memory element could be ARAM or DRAM or any other element hooked up to ACONNECT. It should be configured Memory-to-Memory direction if both source and target are connected ADMA through ACONNECT. 1 = MEMORY_TO_MEMORY 2 = AHUB_TO_MEMORY 4 = MEMORY_TO_AHUB 8 = AHUB_TO_AHUB
10:8	0x1	TRANSFER_MODE: ADMA can be configured to operate in 3 modes 1) ONCE: Enables single transfer that spans single or multiple buffers. Hardware generates an interrupt and clears the "TRANSFER_ENABLE" bit on completion of transfer 2) CONTINUOUS Enables repetitive transfers that spans single or multiple buffers. Hardware reloads SOURCE Address and TARGET Address after completing the number of block transfers indicated by the register field "SOURCE_MEMORY_BUFFERS or TARGET_MEMORY_BUFFERS" and reloads Transfer Count at the end of each block transfer for next transfer. This will be continued till software clears "TRANSFER_ENABLE" bit. 3) Linked List: Enables to operate in Linked list mode using chain of descriptors stored in external/ internal memory for loading SOURCE address, TARGET address, Transfer count and next descriptor address at the end of current block transfer till the descriptors are exhausted or software disables transfers by clearing the "TRANSFER_ENABLE" bit. Hardware fetches the descriptor at the beginning of the current transfer if the next descriptor field is set to a non-zero value or at end of the current transfer if it is found to be "0" in the beginning. 1 = ONCE 2 = CONTINUOUS 4 = LINKED_LIST
2	0x0	TRIGGER_ENABLE: Enables wait on one of the DMA channels interrupts listed in "TRIGGER_SELECT" field to occur for starting the data transfer. 0 = FALSE 1 = TRUE
1	0x1	FLOWCTRL_ENABLE: Enables flow controlled transfers to and/or from the FIFO(s) of AHUB channels specified in "TX_REQUEST_SELECT and RX_REQUEST_SELECT". It is the responsibility of ADMA to ensure enough buffer space is available for transmitting the samples or enough samples are available to read from the channel FIFOs in ADMAIF (AHUB). Source and/or target for DMA transfer should be from AHUB when this bit is set. 0 = FALSE 1 = TRUE
0	0x0	TRANSFER_PAUSE: Stops issuing the new requests and waits for all previously issued DMA requests are completed normally for moving into paused state. Clearing this bit causes resuming the DMA transfer 0 = FALSE 1 = TRUE

23.3.15.8 ADMA_CH<*n*>_CONFIG_0

There are 22 ADMA Configuration registers, one per channel (n = 1 through 22).

Offset: 0x28 + (n – 1) * 0x80 | Read/Write: R/W | Reset: 0x00500001 (0bx000x000x10100000000xxxxxxxx0001)

Bit	Reset	Description
30:28	0x0	SOURCE_MEMORY_BUFFERS: Up to 8 logically partitioned and consecutively located buffers/chunks of ARAM/DRAM of size equal to "TRANSFER_COUNT". Hardware generates an interrupt on completion of every buffer. Software can choose to program more than one block/buffer in all three modes "ONCE, CONTINUOUS and LINKED_LIST". In "CONTINUOUS" mode, all these buffers are read in a circular fashion (0-1-2-3...0-1-2-3) continuously until software clears the "TRANSFER_ENABLE" bit. Transfer terminates in ONCE mode when last buffer is drained and SOURCE ADDR, TARGET ADDR, and TC would be re initialized with the descriptor fields in case of Linked list mode. 0 = BUFFER_1 1 = BUFFERS_2 2 = BUFFERS_3 3 = BUFFERS_4 4 = BUFFERS_5 5 = BUFFERS_6 6 = BUFFERS_7 7 = BUFFERS_8
26:24	0x0	TARGET_MEMORY_BUFFERS: Up to 8 logically partitioned and consecutively located buffers/chunks of ARAM/DRAM of size equal to "TRANSFER_COUNT". Hardwaregenerates an interrupt on completion of every buffer. Software can choose to program more than one block/buffer in all three modes "ONCE, CONTINUOUS and LINKED_LIST". In "CONTINUOUS" mode, all these buffers are written into in a circular fashion (0-1-2-3...0-1-2-3) continuously until software clears the "TRANSFER_ENABLE" bit. Transfer terminates in ONCE mode when last buffer is drained and SOURCE ADDR, TARGET ADDR and TC would be re initialized with the descriptor fields in case of Linked list mode. 0 = BUFFER_1 1 = BUFFERS_2 2 = BUFFERS_3 3 = BUFFERS_4 4 = BUFFERS_5 5 = BUFFERS_6 6 = BUFFERS_7 7 = BUFFERS_8
22:20	0x5	BURST_SIZE: Software initializes this field to an appropriate value based on the type of transfer and bandwidth needs of the client and overall DMA needs of the system. This field is applicable for both reads from Source and writes to Target. ADMA does not support different burst sizes for source and target. 1 = WORD_1 2 = WORDS_2 3 = WORDS_4 4 = WORDS_8 5 = WORDS_16
19:16	0x0	SOURCE_ADDR_WRAP: Address wrap around window for source client. AHUB transfers should wrap on single word. 0 = NO_WRAP 1 = WRAP_ON_1_WORD 2 = WRAP_ON_2_WORDS 3 = WRAP_ON_4_WORDS 4 = WRAP_ON_8_WORDS 5 = WRAP_ON_16_WORDS 6 = WRAP_ON_32_WORDS 7 = WRAP_ON_64_WORDS 8 = WRAP_ON_128_WORDS 9 = WRAP_ON_256_WORDS 10 = WRAP_ON_512_WORDS 11 = WRAP_ON_1K_WORDS
15:12	0x0	TARGET_ADDR_WRAP: Address wrap around window for destination client. AHUB transfers should wrap on single word. 0 = NO_WRAP 1 = WRAP_ON_1_WORD 2 = WRAP_ON_2_WORDS 3 = WRAP_ON_4_WORDS 4 = WRAP_ON_8_WORDS 5 = WRAP_ON_16_WORDS 6 = WRAP_ON_32_WORDS 7 = WRAP_ON_64_WORDS 8 = WRAP_ON_128_WORDS 9 = WRAP_ON_256_WORDS 10 = WRAP_ON_512_WORDS 11 = WRAP_ON_1K_WORDS

Bit	Reset	Description
3:0	0x1	WEIGHT_FOR_WRR: Weight of this channel for weighted round robin arbitration. When this channel get its turn, WRR arbiter selects these many requests from this channel prior to switching to the other channels. Arbiter switches by default to other active channels if this channel has no requests to schedule for arbitration.

23.3.15.9 ADMA_CH<*n*>_AHUB_FIFO_CTRL_0

There are 22 ADMA FIFO Control registers, one per channel (n = 1 through 22).

For n = 1 through 4: Offset: 0x2c + (n – 1) * 0x80 | Read/Write: R/W | Reset: 0x01010303 (0b0xx00001xxx00001xxx00011xxx00011)
For n = 5 through 19: Offset: 0x22c + (n – 5) * 0x80 | Read/Write: R/W | Reset: 0x00000202 (0b0xx00000xxx00000xxx00010xxx00010)
For n = 20 through 22: Offset: 0x9ac + (n – 20) * 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b0xx00000xxx00000xxx00000xxx00000)

Bit	Reset	Description
31	0x0	FETCHING_POLICY: 0: DMA requests are scheduled for arbitration as and when a burst of space is available to load into TX FIFO or a burst of samples are ready to be read from Rx FIFO 1: DMA requests are scheduled for arbitration based on "OVERFLOW and STARVATION" thresholds to let DRAM get into IDLE state for longer durations 0 = BURST_BASED 1 = THRESHOLD_BASED
28:24	0x1	OVERFLOW_THRESHOLD: Burst of Transfers starts when FIFO status reaches this threshold and continues until it becomes empty. Transfer resumes again after FIFO status reaches this threshold. It is applicable only when the FETCHING_POLICY field is set to 1. It must be set to a value in multiples of 16 words.
20:16	0x1	STARVATION_THRESHOLD: Burst of Transfers starts when FIFO count goes below this threshold and continues until it becomes full. Transfer resumes again after FIFO count goes below this threshold. It is applicable only when the FETCHING_POLICY field is set to 1, and It must be set to a value in multiples of 16 words.
12:8	0x3	TX_FIFO_SIZE: TX FIFO size of AHUB channel indicated by the TX_REQUEST_SELECT field in multiples of 16 words/64 bytes. This field is valid only when the FLOWCTRL_ENABLE bit is set and TRANSFER_DIRECTION is either memory-to-AHUB or AHUB-to-AHUB.
4:0	0x3	RX_FIFO_SIZE: RX FIFO size of AHUB channel indicated by the RX_REQUEST_SELECT field in multiples of 16 words/64 bytes. This field is valid only when the FLOWCTRL_ENABLE bit is set and TRANSFER_DIRECTION is either AHUB-to-memory or AHUB-to-AHUB.

23.3.15.10 ADMA_CH<*n*>_TC_STATUS_0

There are 22 ADMA Transfer Count Status registers, one per channel (n = 1 through 22).

Offset: 0x30 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:2	X	CURRENT_TRANSFER_COUNT: Remaining number of words to be transferred

23.3.15.11 ADMA_CH<*n*>_LOWER_SOURCE_ADDR_0

There are 22 ADMA Lower Source Address registers, one per channel (n = 1 through 22).

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	BASE_ADDR: Word aligned lower 32 bits of source address. Next location is reserved for future enhancements to the address width.

23.3.15.12 ADMA_CH<*n*>_LOWER_TARGET_ADDR_0

There are 22 ADMA Lower Target Address registers, one per channel (n = 1 through 22).

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	BASE_ADDR: Word aligned lower 32 bits of target address. Next location is reserved for future enhancements to the address width.

23.3.15.13 ADMA_CH<*n*>_TC_0

There are 22 ADMA Transfer Count registers, one per channel (n = 1 through 22).

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000000000000xx)

Bit	Reset	Description
29:2	0x0	TRANSFER_COUNT: Word aligned transfer size.

23.3.15.14 ADMA_CH<*n*>_LOWER_DESC_ADDR_0

There are 22 ADMA Lower Descriptor registers, one per channel (n = 1 through 22).

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	NEXT_DESCRIPTOR_ADDR: Initially it is loaded by software while setting up the channel and then automatically loaded by hardware prior to end of each block transfer. This is valid only when Linked list mode is enabled. ADMA keeps fetching the descriptor from ARAM/DRAM pointed to by "NEXT_DESCRIPTOR_ADDR" until either software disables the channel or received "NEXT_DESCRIPTOR_ADDR" value is "0". Descriptor is a set of 4 registers (SOURCE_ADDR, TARGET_ADDR, TRANSFER_COUNT and NEXT_DESCRIPTOR_ADDR) located in consecutive 32-bit entries of DRAM/ARAM starting with SOURCE_ADDR.

23.3.15.15 ADMA_CH<*n*>_TRANSFER_STATUS_0

There are 22 ADMA Transfer Status registers, one per channel (n = 1 through 22).

Offset: 0x54 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	TRANSFER_DONE_COUNT: Indicates the number of block/buffer transfers completed since the channel was enabled. In other words the count increments every time the TRANSFER_DONE interrupt is asserted.

23.3.15.16 ADMA_GLOBAL_CMD_0

Offset: 0xc00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	GLOBAL_ENABLE: This bit must be set to "1" for enabling the DMA transfers from channels 0 = FALSE 1 = TRUE

23.3.15.17 ADMA_GLOBAL_SOFT_RESET_0

Offset: 0xc04 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE: Self clearing soft reset 0 = FALSE 1 = TRUE

23.3.15.18 ADMA_GLOBAL_CG_0

Offset: 0xc08 | Read/Write: R/W | Reset: 0x00000007 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx111)

Bit	Reset	Description
2	0x1	CFG_SLCG_ENABLE: Second level clock gating enable for the clock driving the registers 0 = FALSE 1 = TRUE
1	0x1	CHANNEL_SLCG_ENABLE: Second level clock gating enable for the clock driving the channel controllers except of first 4 channels 0 = FALSE 1 = TRUE
0	0x1	GLOBAL_SLCG_ENABLE: Second level clock gating enable for the clock driving first 4 channel controllers and remaining modules 0 = FALSE 1 = TRUE

23.3.15.19 ADMA_GLOBAL_STATUS_0

Offset: 0xc10 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	TRANSFER_PAUSED: Transfers from ALL channels can be paused by setting this bit and can be resumed by clearing it 0 = FALSE 1 = TRUE
3	X	CFG_CLK_ENABLED: Indicates whether Registers clock is enabled or disable 0 = FALSE 1 = TRUE
2	X	CHANNEL_CLK_ENABLED: Indicates whether the clock driving channel controllers is enabled or disabled 0 = FALSE 1 = TRUE
1	X	GLOBAL_CLK_ENABLED: Indicates whether the clock driving common modules is enabled or disabled 0 = FALSE 1 = TRUE
0	X	TRANSFER_ENABLED: Asserts when any of the channels is enabled for DMA transfer 0 = FALSE 1 = TRUE

23.3.15.20 ADMA_GLOBAL_INT_STATUS_0

Offset: 0xc14 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	GLOBAL_TRANSFER_ERROR: Pre mask error interrupt "STATUS" 0 = FALSE 1 = TRUE

23.3.15.21 ADMA_GLOBAL_INT_MASK_0

Offset: 0xc18 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	GLOBAL_TRANSFER_ERROR: Mask for hardware or software triggered error interrupt 0 = FALSE 1 = TRUE

23.3.15.22 ADMA_GLOBAL_INT_SET_0

Offset: 0xc1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	GLOBAL_TRANSFER_ERROR: Self clearing error interrupt "SET" from software. This bit always read back 0 0 = FALSE 1 = TRUE

23.3.15.23 ADMA_GLOBAL_INT_CLEAR_0

Offset: 0xc20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	GLOBAL_TRANSFER_ERROR: Self clearing error interrupt "CLEAR". This bit always read back 0 0 = FALSE 1 = TRUE

23.3.15.24 ADMA_GLOBAL_CTRL_0

Offset: 0xc24 | Read/Write: R/W | Reset: 0x00080800 (0bxxxxxxxxxxxx1000xxxx1000xxxxxxx0)

Bit	Reset	Description
19:16	0x8	OUTSTANDING_MEM_WRITES: Supports a max of 8 outstanding memory reads to ARAM/DRAM
11:8	0x8	OUTSTANDING_MEM_READS: Supports a max of 8 outstanding memory writes to ARAM/DRAM.
0	0x0	TRANSFER_PAUSE: Software may choose to PASUE Transfers from all channels by setting this bit to "1". Clearing it resume all transfers. 0 = DISABLE 1 = ENABLE

23.3.15.25 ADMA_GLOBAL_CH_INT_STATUS_0

Offset: 0xc28 | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21	X	CH22_TRANSFER_DONE: Pre mask end of transfer or transfer done interrupt from all channels 0 = FALSE 1 = TRUE
20	X	CH21_TRANSFER_DONE: 0 = FALSE 1 = TRUE
19	X	CH20_TRANSFER_DONE: 0 = FALSE 1 = TRUE
18	X	CH19_TRANSFER_DONE: 0 = FALSE 1 = TRUE
17	X	CH18_TRANSFER_DONE: 0 = FALSE 1 = TRUE
16	X	CH17_TRANSFER_DONE: 0 = FALSE 1 = TRUE
15	X	CH16_TRANSFER_DONE: 0 = FALSE 1 = TRUE
14	X	CH15_TRANSFER_DONE: 0 = FALSE 1 = TRUE
13	X	CH14_TRANSFER_DONE: 0 = FALSE 1 = TRUE

Bit	Reset	Description
12	X	CH13_TRANSFER_DONE: 0 = FALSE 1 = TRUE
11	X	CH12_TRANSFER_DONE: 0 = FALSE 1 = TRUE
10	X	CH11_TRANSFER_DONE: 0 = FALSE 1 = TRUE
9	X	CH10_TRANSFER_DONE: 0 = FALSE 1 = TRUE
8	X	CH9_TRANSFER_DONE: 0 = FALSE 1 = TRUE
7	X	CH8_TRANSFER_DONE: 0 = FALSE 1 = TRUE
6	X	CH7_TRANSFER_DONE: 0 = FALSE 1 = TRUE
5	X	CH6_TRANSFER_DONE: 0 = FALSE 1 = TRUE
4	X	CH5_TRANSFER_DONE: 0 = FALSE 1 = TRUE
3	X	CH4_TRANSFER_DONE: 0 = FALSE 1 = TRUE
2	X	CH3_TRANSFER_DONE: 0 = FALSE 1 = TRUE
1	X	CH2_TRANSFER_DONE: 0 = FALSE 1 = TRUE
0	X	CH1_TRANSFER_DONE: 0 = FALSE 1 = TRUE

23.3.15.26 ADMA_GLOBAL_CH_ENABLE_STATUS_0

Offset: 0xc2c | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21	X	CH22_TRANSFER_ENABLED: Channel enable status from all channels 0 = FALSE 1 = TRUE
20	X	CH21_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
19	X	CH20_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
18	X	CH19_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
17	X	CH18_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE

Bit	Reset	Description
16	X	CH17_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
15	X	CH16_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
14	X	CH15_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
13	X	CH14_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
12	X	CH13_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
11	X	CH12_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
10	X	CH11_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
9	X	CH10_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
8	X	CH9_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
7	X	CH8_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
6	X	CH7_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
5	X	CH6_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
4	X	CH5_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
3	X	CH4_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
2	X	CH3_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
1	X	CH2_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
0	X	CH1_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE

23.3.15.27 ADMA_GLOBAL_TX_REQUESTORS_0

Status of AHUB transmit channels

Offset: 0xc30 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	AHUB_CH10_ENABLED: 0 = FALSE 1 = TRUE

Bit	Reset	Description
8	X	AHUB_CH9_ENABLED: 0 = FALSE 1 = TRUE
7	X	AHUB_CH8_ENABLED: 0 = FALSE 1 = TRUE
6	X	AHUB_CH7_ENABLED: 0 = FALSE 1 = TRUE
5	X	AHUB_CH6_ENABLED: 0 = FALSE 1 = TRUE
4	X	AHUB_CH5_ENABLED: 0 = FALSE 1 = TRUE
3	X	AHUB_CH4_ENABLED: 0 = FALSE 1 = TRUE
2	X	AHUB_CH3_ENABLED: 0 = FALSE 1 = TRUE
1	X	AHUB_CH2_ENABLED: 0 = FALSE 1 = TRUE
0	X	AHUB_CH1_ENABLED: 0 = FALSE 1 = TRUE

23.3.15.28 ADMA_GLOBAL_RX_REQUESTORS_0

Status of AHUB receive channels

Offset: 0xc34 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	AHUB_CH10_ENABLED: 0 = FALSE 1 = TRUE
8	X	AHUB_CH9_ENABLED: 0 = FALSE 1 = TRUE
7	X	AHUB_CH8_ENABLED: 0 = FALSE 1 = TRUE
6	X	AHUB_CH7_ENABLED: 0 = FALSE 1 = TRUE
5	X	AHUB_CH6_ENABLED: 0 = FALSE 1 = TRUE
4	X	AHUB_CH5_ENABLED: 0 = FALSE 1 = TRUE
3	X	AHUB_CH4_ENABLED: 0 = FALSE 1 = TRUE
2	X	AHUB_CH3_ENABLED: 0 = FALSE 1 = TRUE
1	X	AHUB_CH2_ENABLED: 0 = FALSE 1 = TRUE

Bit	Reset	Description
0	X	AHUB_CH1_ENABLED: 0 = FALSE 1 = TRUE

23.3.15.29 ADMA_GLOBAL_TRIGGERS_0

Status of triggers mapped to the active channels for initiating the transfers

Offset: 0xc38 | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21	X	CH22_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
20	X	CH21_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
19	X	CH20_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
18	X	CH19_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
17	X	CH18_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
16	X	CH17_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
15	X	CH16_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
14	X	CH15_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
13	X	CH14_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
12	X	CH13_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
11	X	CH12_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
10	X	CH11_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
9	X	CH10_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
8	X	CH9_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
7	X	CH8_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
6	X	CH7_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
5	X	CH6_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE

Bit	Reset	Description
4	X	CH5_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
3	X	CH4_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
2	X	CH3_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
1	X	CH2_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE
0	X	CH1_TRIGGER_ASSERTED: 0 = FALSE 1 = TRUE

23.3.15.30 ADMA_GLOBAL_TRANSFER_ERROR_LOG_0

Read/write response status and corresponding channel ID. An interrupt is generated on receiving an error response for read/write request made to ARAM/DRAM.

Offset: 0xc3c | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21:16	X	RID
13:12	X	RRESP: 0 = OKAY 1 = EXOKAY 2 = SLVERR 3 = DECERR
9:4	X	BID
1:0	X	BRESP: 0 = OKAY 1 = EXOKAY 2 = SLVERR 3 = DECERR

23.3.16 ADMAIF AXBAR Registers

23.3.16.1 ADMAIF_AXBAR_RX<n>_ENABLE_0

There are 10 ADMAIF AXBAR RX Enable registers, one for each RX channel (n=1 to 10).

Offset: 0x0 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE: 0 = FALSE 1 = TRUE

23.3.16.2 ADMAIF_AXBAR_RX<n>_SOFT_RESET_0

There are 10 ADMAIF AXBAR RX Soft Reset registers, one for each RX channel (n=1 to 10).

Offset: 0x4 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.16.3 ADMAIF_AXBAR_RX<n>_STATUS_0

There are 10 ADMAIF AXBAR RX Status registers, one for each RX channel (n=1 to 10).

Offset: 0xc + (n – 1) * 0x40 | Read/Write: RO | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	AVAILABLE_CREDITS
7	X	DMA_FIFO_FULL: 0 = FALSE 1 = TRUE
6	X	DMA_FIFO_EMPTY: 0 = FALSE 1 = TRUE
5	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
4	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	TRANSFER_ENABLED: 0 = FALSE 1 = TRUE

23.3.16.4 ADMAIF_AXBAR_RX<n>_INT_STATUS_0

There are 10 ADMAIF AXBAR RX Interrupt Status registers, one for each RX channel (n=1 to 10).

Offset: 0x10 + (n – 1) * 0x40 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	DMA_OVERRUN: 0 = FALSE 1 = TRUE

23.3.16.5 ADMAIF_AXBAR_RX<n>_INT_MASK_0

There are 10 ADMAIF AXBAR RX Interrupt Mask registers, one for each RX channel (n=1 to 10).

Offset: 0x14 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	DMA_OVERRUN: 0 = UNMASK 1 = MASK

23.3.16.6 ADMAIF_AXBAR_RX<n>_INT_SET_0

There are 10 ADMAIF AXBAR RX Interrupt Set registers, one for each RX channel (n=1 to 10).

Offset: 0x18 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	DMA_OVERRUN: 0 = FALSE 1 = TRUE

23.3.16.7 ADMAIF_AXBAR_RX<n>_INT_CLEAR_0

There are 10 ADMAIF AXBAR RX Interrupt Clear registers, one for each RX channel (n=1 to 10).

Offset: 0x1c + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	DMA_OVERRUN: 0 = FALSE 1 = TRUE

23.3.16.8 ADMAIF_AXBAR_RX<n>_CIF_CTRL_0

There are 10 ADMAIF AXBAR RX CIF Control registers, one for each RX channel (n=1 to 10).

Offset: 0x20 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00007700 (0b0000000000000000x111x1110000x000)

Bit	Reset	Description
31	0x0	PACK8_ENABLE
30	0x0	PACK16_ENABLE
29:24	0x0	FIFO_THRESHOLD
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	Reset	Description
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.16.9 ADMAIF_AXBAR_RX<n>_FIFO_CTRL_0

There are 10 ADMAIF AXBAR RX FIFO Control registers, one for each RX channel (n=1 to 10). Each channel's FIFO Control register has a unique reset value.

RX1: Offset: 0x28 | Read/Write: R/W | Reset: 0x00000300 (0b0xxxxxxxxxxxxxxxxxx00011xxx00000)
RX2: Offset: 0x68 | Read/Write: R/W | Reset: 0x00000304 (0b0xxxxxxxxxxxxxxxxxx00011xxx00100)
RX3: Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000208 (0b0xxxxxxxxxxxxxxxxxx00010xxx01000)
RX4: Offset: 0xe8 | Read/Write: R/W | Reset: 0x0000020b (0b0xxxxxxxxxxxxxxxxxx00010xxx01011)
RX5: Offset: 0x128 | Read/Write: R/W | Reset: 0x0000020e (0b0xxxxxxxxxxxxxxxxxx00010xxx01110)
RX6: Offset: 0x168 | Read/Write: R/W | Reset: 0x00000211 (0b0xxxxxxxxxxxxxxxxxx00010xxx10001)
RX7: Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000214 (0b0xxxxxxxxxxxxxxxxxx00010xxx10100)
RX8: Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000217 (0b0xxxxxxxxxxxxxxxxxx00010xxx10111)
RX9: Offset: 0x228 | Read/Write: R/W | Reset: 0x0000021a (0b0xxxxxxxxxxxxxxxxxx00010xxx11010)
RX10: Offset: 0x268 | Read/Write: R/W | Reset: 0x0000021d (0b0xxxxxxxxxxxxxxxxxx00010xxx11101)

Bit	Reset	Description
31	Depends on RX<n>. See reset values above this table.	TRANSFER_MODE: 0 = DMA 1 = PIO
12:8		DMA_FIFO_SIZE
4:0		DMA_FIFO_START_ADDR

23.3.16.10 ADMAIF_AXBAR_RX<n>_FIFO_READ_0

There are 10 ADMAIF AXBAR RX FIFO Read registers, one for each RX channel (n=1 to 10).

Offset: 0x2c + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

23.3.16.11 ADMAIF_AXBAR_TX<n>_ENABLE_0

There are 10 ADMAIF AXBAR TX Enable registers, one for each TX channel (n=1 to 10).

Offset: 0x300 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE: 0 = FALSE 1 = TRUE

23.3.16.12 ADMAIF_AXBAR_TX<n>_SOFT_RESET_0

There are 10 ADMAIF AXBAR TX Soft Reset registers, one for each TX channel (n=1 to 10).

Offset: 0x304 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.16.13 ADMAIF_AXBAR_TX<n>_STATUS_0

There are 10 ADMAIF AXBAR TX Status registers, one for each TX channel (n=1 to 10).

Offset: 0x30c + (n – 1) * 0x40 | Read/Write: RO | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	AVAILABLE_CREDITS
7	X	DMA_FIFO_FULL: 0 = FALSE 1 = TRUE
6	X	DMA_FIFO_EMPTY: 0 = FALSE 1 = TRUE
5	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
4	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	TRANSFER_ENABLED: 0 = FALSE 1 = TRUE

23.3.16.14 ADMAIF_AXBAR_TX<n>_INT_STATUS_0

There are 10 ADMAIF AXBAR TX Interrupt Status registers, one for each TX channel (n=1 to 10).

Offset: 0x310 + (n – 1) * 0x40 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	DMA_UNDERRUN: 0 = FALSE 1 = TRUE

23.3.16.15 ADMAIF_AXBAR_TX<n>_INT_MASK_0

There are 10 ADMAIF AXBAR TX Interrupt Mask registers, one for each TX channel (n=1 to 10).

Offset: 0x314 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	DMA_UNDERRUN: 0 = UNMASK 1 = MASK

23.3.16.16 ADMAIF_AXBAR_TX<n>_INT_SET_0

There are 10 ADMAIF AXBAR TX Interrupt Set registers, one for each TX channel (n=1 to 10).

Offset: 0x318 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	DMA_UNDERRUN: 0 = FALSE 1 = TRUE

23.3.16.17 ADMAIF_AXBAR_TX<n>_INT_CLEAR_0

There are 10 ADMAIF AXBAR TX Interrupt Clear registers, one for each TX channel (n=1 to 10).

Offset: 0x31c + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	DMA_UNDERRUN: 0 = FALSE 1 = TRUE

23.3.16.18 ADMAIF_AXBAR_TX<n>_CIF_CTRL_0

There are 10 ADMAIF AXBAR TX CIF Control registers, one for each TX channel (n=1 to 10).

Offset: 0x320 + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00007700 (0b0000000000000000x111x1110000x000)

Bit	Reset	Description
31	0x0	UNPACK8_ENABLE
30	0x0	UNPACK16_ENABLE
29:24	0x0	FIFO_THRESHOLD
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16

Bit	Reset	Description
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.16.19 ADMAIF_AXBAR_TX<n>_FIFO_CTRL_0

There are 10 ADMAIF AXBAR TX FIFO Control registers, one for each TX channel (n=1 to 10). Each channel's FIFO Control register has a unique reset value.

TX1: Offset: 0x328 | Read/Write: R/W | Reset: 0x02000300 (0b0x0000100000xxxxxxx00011xxx00000)
TX2: Offset: 0x368 | Read/Write: R/W | Reset: 0x02000304 (0b0x0000100000xxxxxxx00011xxx00100)
TX3: Offset: 0x3a8 | Read/Write: R/W | Reset: 0x01800208 (0b0x0000011000xxxxxxx00010xxx01000)
TX4: Offset: 0x3e8 | Read/Write: R/W | Reset: 0x0180020b (0b0x0000011000xxxxxxx00010xxx01011)
TX5: Offset: 0x428 | Read/Write: R/W | Reset: 0x0180020e (0b0x0000011000xxxxxxx00010xxx01110)
TX6: Offset: 0x468 | Read/Write: R/W | Reset: 0x01800211 (0b0x0000011000xxxxxxx00010xxx10001)
TX7: Offset: 0x4a8 | Read/Write: R/W | Reset: 0x01800214 (0b0x0000011000xxxxxxx00010xxx10100)
TX8: Offset: 0x4e8 | Read/Write: R/W | Reset: 0x01800217 (0b0x0000011000xxxxxxx00010xxx10111)
TX9: Offset: 0x528 | Read/Write: R/W | Reset: 0x0180021a (0b0x0000011000xxxxxxx00010xxx11010)
TX10: Offset: 0x568 | Read/Write: R/W | Reset: 0x0180021d (0b0x0000011000xxxxxxx00010xxx11101)

Bit	Reset	Description
31	Depends on TX<n>. See reset values above this table.	TRANSFER_MODE: 0 = DMA 1 = PIO
29:20		DMA_FIFO_THRESHOLD
12:8		DMA_FIFO_SIZE
4:0		DMA_FIFO_START_ADDR

23.3.16.20 ADMAIF_AXBAR_TX<n>_FIFO_WRITE_0

There are 10 ADMAIF AXBAR TX FIFO Write registers, one for each TX channel (n=1 to 10).

Offset: 0x32c + (n – 1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.3.16.21 ADMAIF_GLOBAL_ENABLE_0

Offset: 0x700 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE: 0 = FALSE 1 = TRUE

23.3.16.22 ADMAIF_GLOBAL_SOFT_RESET_0

Self-clearing software-initiated global reset. It clears all channels' enable bits, DMA/CIF FIFOs, state machines, and any pipelines.

Offset: 0x704 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.16.23 ADMAIF_GLOBAL_CG_0

Offset: 0x708 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1	0x1	CHANNEL_SLCG_ENABLE: Second level clock gating enable, for remaining channels specific logic portions. 0 = FALSE 1 = TRUE
0	0x1	GLOBAL_SLCG_ENABLE: Second level clock gating enable for first 2 channels and global/common logic. 0 = FALSE 1 = TRUE

23.3.16.24 ADMAIF_GLOBAL_STATUS_0

Offset: 0x710 | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CHANNEL_CLK_ENABLE: Channel clock Enable 0 = FALSE 1 = TRUE
8	X	GLOBAL_CLK_ENABLE: Global clock Enable 0 = FALSE 1 = TRUE
0	X	TRANSFER_ENABLED: Set when at least one of the channels is enabled by software 0 = FALSE 1 = TRUE

23.3.16.25 ADMAIF_GLOBAL_RX_ENABLE_STATUS_0

Offset: 0x720 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
8	X	CH9_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
7	X	CH8_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
6	X	CH7_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
5	X	CH6_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
4	X	CH5_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
3	X	CH4_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
2	X	CH3_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
1	X	CH2_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
0	X	CH1_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE

23.3.16.26 ADMAIF_GLOBAL_TX_ENABLE_STATUS_0

Offset: 0x724 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
8	X	CH9_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
7	X	CH8_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
6	X	CH7_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
5	X	CH6_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
4	X	CH5_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
3	X	CH4_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE

Bit	Reset	Description
2	X	CH3_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
1	X	CH2_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE
0	X	CH1_TRANSFER_ENABLED: 0 = FALSE 1 = TRUE

23.3.16.27 ADMAIF_GLOBAL_RX_DMA_FIFO_EMPTY_STATUS_0

Offset: 0x728 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_EMPTY: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_EMPTY: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_EMPTY: 0 = FALSE 1 = TRUE
6	X	CH7_FIFO_EMPTY: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_EMPTY: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_EMPTY: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_EMPTY: 0 = FALSE 1 = TRUE
2	X	CH3_FIFO_EMPTY: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.16.28 ADMAIF_GLOBAL_RX_DMA_FIFO_FULL_STATUS_0

Offset: 0x72c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_FULL: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_FULL: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_FULL: 0 = FALSE 1 = TRUE

Bit	Reset	Description
6	X	CH7_FIFO_FULL: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_FULL: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_FULL: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_FULL: 0 = FALSE 1 = TRUE
2	X	CH3_FIFO_FULL: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_FULL: 0 = FALSE 1 = TRUE

23.3.16.29 ADMAIF_GLOBAL_TX_DMA_FIFO_EMPTY_STATUS_0

Offset: 0x730 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_EMPTY: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_EMPTY: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_EMPTY: 0 = FALSE 1 = TRUE
6	X	CH7_FIFO_EMPTY: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_EMPTY: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_EMPTY: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_EMPTY: 0 = FALSE 1 = TRUE
2	X	CH3_FIFO_EMPTY: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.16.30 ADMAIF_GLOBAL_TX_DMA_FIFO_FULL_STATUS_0

Offset: 0x734 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_FULL: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_FULL: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_FULL: 0 = FALSE 1 = TRUE
6	X	CH7_FIFO_FULL: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_FULL: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_FULL: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_FULL: 0 = FALSE 1 = TRUE
2	X	CH3_FIFO_FULL: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_FULL: 0 = FALSE 1 = TRUE

23.3.16.31 ADMAIF_GLOBAL_RX_ACIF_FIFO_EMPTY_STATUS_0

Offset: 0x738 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_EMPTY: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_EMPTY: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_EMPTY: 0 = FALSE 1 = TRUE
6	X	CH7_FIFO_EMPTY: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_EMPTY: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_EMPTY: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_EMPTY: 0 = FALSE 1 = TRUE

Bit	Reset	Description
2	X	CH3_FIFO_EMPTY: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.16.32 ADMAIF_GLOBAL_RX_ACIF_FIFO_FULL_STATUS_0

Offset: 0x73c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_FULL: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_FULL: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_FULL: 0 = FALSE 1 = TRUE
6	X	CH7_FIFO_FULL: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_FULL: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_FULL: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_FULL: 0 = FALSE 1 = TRUE
2	X	CH3_FIFO_FULL: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_FULL: 0 = FALSE 1 = TRUE

23.3.16.33 ADMAIF_GLOBAL_TX_ACIF_FIFO_EMPTY_STATUS_0

Offset: 0x740 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_EMPTY: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_EMPTY: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_EMPTY: 0 = FALSE 1 = TRUE

Bit	Reset	Description
6	X	CH7_FIFO_EMPTY: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_EMPTY: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_EMPTY: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_EMPTY: 0 = FALSE 1 = TRUE
2	X	CH3_FIFO_EMPTY: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.16.34 ADMAIF_GLOBAL_TX_ACIF_FIFO_FULL_STATUS_0

Offset: 0x744 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_FIFO_FULL: 0 = FALSE 1 = TRUE
8	X	CH9_FIFO_FULL: 0 = FALSE 1 = TRUE
7	X	CH8_FIFO_FULL: 0 = FALSE 1 = TRUE
6	X	CH7_FIFO_FULL: 0 = FALSE 1 = TRUE
5	X	CH6_FIFO_FULL: 0 = FALSE 1 = TRUE
4	X	CH5_FIFO_FULL: 0 = FALSE 1 = TRUE
3	X	CH4_FIFO_FULL: 0 = FALSE 1 = TRUE
2	X	CH3_FIFO_FULL: 0 = FALSE 1 = TRUE
1	X	CH2_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	CH1_FIFO_FULL: 0 = FALSE 1 = TRUE

23.3.16.35 ADMAIF_GLOBAL_RX_DMA_OVERRUN_INT_STATUS_0

Offset: 0x748 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_DMA_OVERRUN: 0 = FALSE 1 = TRUE
8	X	CH9_DMA_OVERRUN: 0 = FALSE 1 = TRUE
7	X	CH8_DMA_OVERRUN: 0 = FALSE 1 = TRUE
6	X	CH7_DMA_OVERRUN: 0 = FALSE 1 = TRUE
5	X	CH6_DMA_OVERRUN: 0 = FALSE 1 = TRUE
4	X	CH5_DMA_OVERRUN: 0 = FALSE 1 = TRUE
3	X	CH4_DMA_OVERRUN: 0 = FALSE 1 = TRUE
2	X	CH3_DMA_OVERRUN: 0 = FALSE 1 = TRUE
1	X	CH2_DMA_OVERRUN: 0 = FALSE 1 = TRUE
0	X	CH1_DMA_OVERRUN: 0 = FALSE 1 = TRUE

23.3.16.36 ADMAIF_GLOBAL_TX_DMA_UNDERRUN_INT_STATUS_0

Offset: 0x74c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH10_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
8	X	CH9_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
7	X	CH8_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
6	X	CH7_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
5	X	CH6_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
4	X	CH5_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
3	X	CH4_DMA_UNDERRUN: 0 = FALSE 1 = TRUE

Bit	Reset	Description
2	X	CH3_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
1	X	CH2_DMA_UNDERRUN: 0 = FALSE 1 = TRUE
0	X	CH1_DMA_UNDERRUN: 0 = FALSE 1 = TRUE

23.3.16.37 ADMAIF_GLOBAL_CYA_0

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x750..0x757 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CYA

23.3.16.38 ADMAIF_GLOBAL_DBG_0

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x758..0x75f | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

23.3.17 AGIC Registers

23.3.17.1 AGIC_DISTRIBUTOR_GICD_CTLR_0

Offset: 0x1000 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_CTLR: Distributor Control Register

23.3.17.2 AGIC_DISTRIBUTOR_GICD_TYPER_0

Offset: 0x1004 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_TYPER: Interrupt Controller Type Register

23.3.17.3 AGIC_DISTRIBUTOR_GICD_IIDR_0

Offset: 0x1008 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_IIDR: Distributor Implementer Identification Register

23.3.17.4 AGIC_DISTRIBUTOR_GICD_IGROUPR<n>_0

There are 16 GICD Interrupt Group registers, where n = 0 to 15.

Offset: 0x1080 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_IGROUPR<n>: Interrupt Group Registers

23.3.17.5 AGIC_DISTRIBUTOR_GICD_ISENABLER<n>_0

There are 16 GICD Interrupt Set-Enable registers, where n = 0 to 15.

Offset: 0x1100 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ISENABLER<n>: Interrupt Set-Enable Registers

23.3.17.6 AGIC_DISTRIBUTOR_GICD_ICENABLER<n>_0

There are 16 GICD Interrupt Clear-Enable registers, where n = 0 to 15.

Offset: 0x1180 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ICENABLER<n>: Interrupt Clear-Enable Registers

23.3.17.7 AGIC_DISTRIBUTOR_GICD_ISPENDR<n>_0

There are 16 GICD Interrupt Set-Pending registers, where n = 0 to 15.

Offset: 0x1200 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ISPENDR<n>: Interrupt Set-Pending Registers

23.3.17.8 AGIC_DISTRIBUTOR_GICD_ICPENDR<n>_0

There are 16 GICD Interrupt Clear-Pending registers, where n = 0 to 15.

Offset: 0x1280 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ICPENDR<n>: Interrupt Clear-Pending Registers

23.3.17.9 AGIC_DISTRIBUTOR_GICD_ISACTIVER<n>_0

There are 16 GICD Interrupt Set-Active registers, where n = 0 to 15.

Offset: 0x1300 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ISACTIVER<n>: Interrupt Set-Active Registers

23.3.17.10 AGIC_DISTRIBUTOR_GICD_ICACTIVER<n>_0

There are 16 GICD Interrupt Clear-Active registers, where n = 0 to 15.

Offset: 0x1380 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ICACTIVER<n>: Interrupt Clear-Active Registers

23.3.17.11 AGIC_DISTRIBUTOR_GICD_IPRIORITYR<n>_0

There are 128 GICD Interrupt Priority registers, where n = 0 to 127.

Offset: 0x1400 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_IPRIORITYR<n>: Interrupt Priority Registers

23.3.17.12 AGIC_DISTRIBUTOR_GICD_ITARGETSR<n>_0

There are 128 GICD Interrupt Processor Targets registers, where n = 0 to 127.

Offset: 0x1800 + (n * 0x4) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ITARGETSR<n>: Interrupt Processor Targets Registers

23.3.17.13 AGIC_DISTRIBUTOR_GICD_ICFGR<n>_0

There are 32 GICD Interrupt Configuration registers, where n = 0 to 31.

Offset: 0x1c00 + (n * 0x4) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_ICFGR<n>: Interrupt Configuration Registers

23.3.17.14 AGIC_DISTRIBUTOR_GICD_PPISR_0

Offset: 0x1d00 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_PPISR: Private Peripheral Interrupt Status Register

23.3.17.15 AGIC_DISTRIBUTOR_GICD_SPISR<n>_0

There are 15 GICD Shared Peripheral Interrupt Status registers, where n = 0 to 14.

Offset: 0x1d04 + (n * 0x4) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_SPISR<n>: Shared Peripheral Interrupt Status Registers

23.3.17.16 AGIC_DISTRIBUTOR_GICD_SGIR_0

Offset: 0x1f00 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_SGIR: Software Generated Interrupt Register

23.3.17.17 AGIC_DISTRIBUTOR_GICD_CPENDSGIR<n>_0

There are 4 GICD SGI Clear-Pending registers, where n = 0 to 3.

Offset: 0x1f10 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_CPENDSGIR<n>: SGI Clear-Pending Registers

23.3.17.18 AGIC_DISTRIBUTOR_GICD_SPENDSGIR<n>_0

There are 4 GICD SGI Set-Pending registers, where n = 0 to 3.

Offset: 0x1f20 + (n * 0x4) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_SPENDSGIR<n>: SGI Set-Pending Registers

23.3.17.19 AGIC_DISTRIBUTOR_GICD_PIDR<n>_0

There are 8 GICD Peripheral ID registers, where n = 0 to 7. Note that ID5 through ID7 register offsets precede those offsets of registers ID0 through ID4.

ID0-ID4: Offset: 0x1fe0 + (n * 0x4) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
ID5-ID7: Offset: 0x1fd0 + (n-5) * 0x4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_PIDR<n>: Peripheral ID<n> Register

23.3.17.20 AGIC_DISTRIBUTOR_GICD_CIDR<n>_0

There are 4 GICD Component ID registers, where n = 0 to 3.

Offset: 0x1ff0 + (n * 0x4) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICD_CIDR<n>: Component ID<n> Register

23.3.17.21 AGIC_CPUIF_GICC_CTLR_0

Offset: 0x2000 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_CTLR: CPU Interface Control Register

23.3.17.22 AGIC_CPUIF_GICC_PMR_0

Offset: 0x2004 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_PMR: Interrupt Priority Mask Register

23.3.17.23 AGIC_CPUIF_GICC_BPR_0

Offset: 0x2008 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_BPR: Binary Point Register

23.3.17.24 AGIC_CPUIF_GICC_IAR_0

Offset: 0x200c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_IAR: Interrupt Acknowledge Register

23.3.17.25 AGIC_CPUIF_GICC_EOIR_0

Offset: 0x2010 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_EOIR: End of Interrupt Register

23.3.17.26 AGIC_CPUIF_GICC_RPR_0

Offset: 0x2014 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_RPR: Running Priority Register

23.3.17.27 AGIC_CPUIF_GICC_HPPIR_0

Offset: 0x2018 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_HPPIR: Highest Priority Pending Interrupt Register

23.3.17.28 AGIC_CPUIF_GICC_ABPR_0

Offset: 0x201c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_ABPR: Aliased Binary Point Register

23.3.17.29 AGIC_CPUIF_GICC_AIAR_0

Offset: 0x2020 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_AIAR: Aliased Interrupt Acknowledge Register

23.3.17.30 AGIC_CPUIF_GICC_AEOIR_0

Offset: 0x2024 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_AEOIR: Aliased End of Interrupt Register

23.3.17.31 AGIC_CPUIF_GICC_AHPPIR_0

Offset: 0x2028 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_AHPPIR: Aliased Highest Priority Pending Interrupt Register

23.3.17.32 AGIC_CPUIF_GICC_APR0_0

Offset: 0x20d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_APR0: Active Priority Register

23.3.17.33 AGIC_CPUIF_GICC_NSAPR0_0

Offset: 0x20e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_NSAPR0: Non-Secure Active Priority Register

23.3.17.34 AGIC_CPUIF_GICC_IIDR_0

Offset: 0x20fc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_IIDR: CPU Interface Identification Register

23.3.17.35 AGIC_CPUIF_GICC_DIR_0

Offset: 0x3000 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_DIR: Deactivate Interrupt Register

23.3.18 Mixer Registers

23.3.18.1 MIXER_AXBAR_RX<n>_SOFT_RESET_0

There are 10 RX Soft Reset registers, where n = 1 through 10.

Offset: 0x4 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.18.2 MIXER_AXBAR_RX<n>_STATUS_0

There are 10 RX Status registers, where n = 1 through 10.

Offset: 0x10 + (n-1) * 0x40 | Read/Write: RO | Reset: 0xX0000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	CONFIG_BUSY: 0 = FALSE 1 = TRUE
1	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
0	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE

23.3.18.3 MIXER_AXBAR_RX<n>_CIF_CTRL_0

There are 10 RX CIF Control registers, where n = 1 through 10.

Offset: 0x24 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00007700 (0bxx00000000000000x111x1110000x000)

Bit	Reset	Description
29:24	0x0	FIFO_THRESHOLD

Bit	Reset	Description
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
2	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	Reset	Description
0	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.18.4 MIXER_AXBAR_RX<n>_CTRL_0

There are 10 RX Control registers, where n = 1 through 10.

Offset: 0x28 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00010823 (0b0xxxxxx0xxx00001000010000010011)

Bit	Reset	Description
24	FALSE	SAMPLE_COUNTER_RESET: 0 = FALSE 1 = TRUE
20:16	0x1	ACT_THRESHOLD
15:0	0x823	DEACT_THRESHOLD

23.3.18.5 MIXER_AXBAR_RX<n>_PEAK_CTRL_0

There are 10 RX Peak Control registers, where n = 1 through 10.

Offset: 0x2c + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x000012c0 (0b00000000000000000001001011000000)

Bit	Reset	Description
31	WINDOW_BASED	PEAK_TYPE: 0 = WINDOW_BASED 1 = RESET_ON_READ
30:0	0x12c0	PEAK_WIN

23.3.18.6 MIXER_AXBAR_RX<n>_SAMPLE_COUNT_0

There are 10 RX Sample Count registers, where n = 1 through 10.

Offset: 0x30 + (n-1) * 0x40 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SAMPLE_CNT

23.3.18.7 MIXER_AXBAR_TX<n>_ENABLE_0

There are 5 TX Enable registers, where n = 1 through 5.

Offset: 0x280 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: 0 = FALSE 1 = TRUE

23.3.18.8 MIXER_AXBAR_TX<n>_SOFT_RESET_0

There are 5 TX Soft Reset registers, where n = 1 through 5.

Offset: 0x284 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: 0 = FALSE 1 = TRUE

23.3.18.9 MIXER_AXBAR_TX<n>_STATUS_0

There are 5 TX Status registers, where n = 1 through 5.

Offset: 0x290 + (n-1) * 0x40 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	ACIF_FIFO_FULL: 0 = FALSE 1 = TRUE
1	X	ACIF_FIFO_EMPTY: 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: 0 = FALSE 1 = TRUE

23.3.18.10 MIXER_AXBAR_TX<n>_INT_STATUS_0

There are 5 TX Interrupt Status registers, where n = 1 through 5.

Offset: 0x294 + (n-1) * 0x40 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TX_DONE: 0 = CLEAR 1 = SET

23.3.18.11 MIXER_AXBAR_TX<n>_INT_MASK_0

There are 5 TX Interrupt Mask registers, where n = 1 through 5.

Offset: 0x298 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	MASK	TX_DONE: 0 = UNMASK 1 = MASK

23.3.18.12 MIXER_AXBAR_TX<n>_INT_SET_0

There are 5 TX Interrupt Set registers, where n = 1 through 5.

Offset: 0x29c + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: 0 = CLEAR 1 = SET

23.3.18.13 MIXER_AXBAR_TX<n>_INT_CLEAR_0

There are 5 TX Interrupt Clear registers, where n = 1 through 5.

Offset: 0x2a0 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	CLEAR	TX_DONE: 0 = CLEAR 1 = SET

23.3.18.14 MIXER_AXBAR_TX<n>_CIF_CTRL_0

There are 5 TX CIF Control registers, where n = 1 through 5.

Offset: 0x2a4 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00007700 (0bxx00000000000000x111x1110000x000)

Bit	Reset	Description
29:24	0x0	FIFO_THRESHOLD
23:20	0x0	AXBAR_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	0x7	AXBAR_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	0x7	CLIENT_BITS: 0 = RVSD 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD

Bit	Reset	Description
2	0x0	REPLICATE: 0 = FALSE 1 = TRUE
1	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	0x0	MONO_CONV: 0 = ZERO 1 = COPY

23.3.18.15 MIXER_AXBAR_TX<n>_ADDER_CONFIG_0

There are 5 TX Adder Configuration registers, where n = 1 through 5.

Offset: 0x2a8 + (n-1) * 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	DISABLE	RX10_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
8	DISABLE	RX9_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
7	DISABLE	RX8_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
6	DISABLE	RX7_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
5	DISABLE	RX6_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
4	DISABLE	RX5_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
3	DISABLE	RX4_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
2	DISABLE	RX3_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
1	DISABLE	RX2_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE
0	DISABLE	RX1_INPUT_ENABLE: 0 = DISABLE 1 = ENABLE

23.3.18.16 MIXER_ENABLE_0

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	ENABLE: MIXER global enable bit, When disabled, the MIXER does not disable until all the data in the datapipe has been processed and sent out. The "datapipe" here includes the mixer core block and TXCIF, but not the RXCIF. The MIXER core will finish the current frame and go bank to IDLE. It will not pop data from the RXCIF any more, even if there are still some data in the RXCIF FIFO. Read back of the register will also give module enable status. 0 = FALSE 1 = TRUE

23.3.18.17 MIXER_SOFT_RESET_0

Offset: 0x404 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SOFT_RESET: Soft Reset is a self-clearing bit. Soft Reset resets all FSM, flushes flow control FIFO, and resets the state registers. It also brings the module back to the disabled state (without flushing data in the pipe) 0 = FALSE 1 = TRUE

23.3.18.18 MIXER_CG_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	TRUE	SLCG_EN: Second level clock gating enable 0 = FALSE 1 = TRUE

23.3.18.19 MIXER_STATUS_0

Offset: 0x410 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	SLCG_CLKEN: 0 = FALSE 1 = TRUE
0	X	ENABLE_STATUS: 0 = FALSE 1 = TRUE

23.3.18.20 MIXER_INT_STATUS_0

Offset: 0x414 | Read/Write: RO | Reset: 0x00XX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	X	TX5_DONE: 0 = CLEAR 1 = SET
19	X	TX4_DONE: 0 = CLEAR 1 = SET
18	X	TX3_DONE: 0 = CLEAR 1 = SET
17	X	TX2_DONE: 0 = CLEAR 1 = SET
16	X	TX1_DONE: 0 = CLEAR 1 = SET
0	X	MIXER_TX_DONE: 0 = CLEAR 1 = SET

23.3.18.21 MIXER_AHUBRAMCTL_GAIN_CONFIG_RAM_CTRL_0

Mixer Configuration RAM

1. The content in the RAM is shown below
2. The order is input stream 0 settings, input stream 1 settings, input stream 9 settings.

Input Stream 0:

0x00: cn_p1_0

0x01: cn_p1_1

0x02: cn_p1_2

0x03: cn_p2_0

0x04: cn_p2_1

0x05: cn_p2_2

0x06: cn_p3_0

0x07: cn_p3_1

0x08: cn_p3_2

0x09: gain_value

0x0a: duration_n1

0x0b: duration_n2

0x0c: duration_n3

0x0d: duration_inv_n3

0x0e: Reserved

0x0f: CONFIG_DONE_TRIGGER

3. For each time of programming, after software programs part or all the parameters which are needed for each stream, software must write CONFIG_DONE_TRIGGER, which offset is 0xf (the last address for each stream). Hardware will trigger the switching after CONFIG_DONE_TRIGGER was programmed. Software can program any value for this register.
4. Software needs to check configuration busy flag before programming this configuration RAM. It can program only if it is not "busy". A Configure Error interrupt will be generated if software programs when it is "busy".
5. Software needs to program the right address for different input streams; that is, 0x0 for input stream0, 0x10 for input stream 1, and 0x90 for input stream 9.
6. Gain_value:
   a. Software can program the settings only if the gain value needs change.
   b. Q16.16 format, 0x10000 means unity gain.

   Hardware running with unity gain as default. Software does not need to program anything if applying unity gain to input stream.
7. This RAM cannot be read by software during the runtime (MIXER is enabling), read port is occupied by hardware
8. Duration_n3:
   a. ==1: doing gain transition (from old gain to new gain) instantaneously.
   b. Should be greater than 64.
9. Duration parameters setting: duration_n1<=duration_n2<=duration_n3.
10. It is meaningless to read RAM for each input stream with offset equal to 0x0e and 0x0f. Hardware may return an unexpected value for these two entries access.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x42c..0x42f | Read/Write: R/W | Reset: 0xX0004000 (0bxxxxxxxx00000000x100xxx000000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
23:16	RW	0x0	SEQ_READ_COUNT
14	RW	WRITE	RW: 0 = READ 1 = WRITE
13	RW	DISABLE	ADDR_INIT_EN: 0 = DISABLE 1 = ENABLE
12	RW	DISABLE	SEQ_ACCESS_EN: 0 = DISABLE 1 = ENABLE
8:0	RW	0x0	RAM_ADDR

23.3.18.22 MIXER_AHUBRAMCTL_GAIN_CONFIG_RAM_DATA_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x430..0x433 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.3.18.23 MIXER_AHUBRAMCTL_PEAKM_RAM_CTRL_0

Peak Metering RAM

1. For Peak value storing
2. The content order is as follows:
   - 0x00: in0_ch0 input stream 0, channel 0
   - 0x01: in0_ch1
   - 0x02: in0_ch2
   - 0x03: in0_ch3
   - 0x04: in0_ch4
   - 0x05: in0_ch5
   - 0x06: in0_ch6
   - 0x07: in0_ch7
   - 0x4e: in9_ch6
   - 0x4f: in9_ch7
3. Peak Metering RAM is read-only for software.
4. Peak Metering RAM supports sequential reads. Software needs to program CTRL.SEQ_READ_COUNT to indicate the number of sequential reads from initial address, that is, if software wants to sequentially read peak values for 8 channels, SEQ_READ_COUNT should be set with 8. SEQ_READ_COUNT is only valid in sequential access mode (SEQ_ACCESS_EN == 1).
5. CTRL.SEQ_READ_COUNT is 8 bits, which means the maximum number of sequential read entries is 255. For more than 255 sequential read entries, software needs to reload this field, which means programming CTRL registers again.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x434..0x437 | Read/Write: R/W | Reset: 0xX0004000 (0bxxxxxxxx00000000x100xxx000000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
23:16	RW	0x0	SEQ_READ_COUNT
14	RW	WRITE	RW: 0 = READ 1 = WRITE
13	RW	DISABLE	ADDR_INIT_EN: 0 = DISABLE 1 = ENABLE
12	RW	DISABLE	SEQ_ACCESS_EN: 0 = DISABLE 1 = ENABLE
8:0	RW	0x0	RAM_ADDR

23.3.18.24 MIXER_AHUBRAMCTL_PEAKM_RAM_DATA_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x438..0x43b | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.3.19 MBDRC Registers

23.3.19.1 MBDRC_CG_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	FALSE	SLCG_EN: Second level clock gating enable 0 = FALSE 1 = TRUE

23.3.19.2 MBDRC_STATUS_0

Offset: 0xc | Read/Write: RO | Reset: 0xX000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SLCG_CLKEN: 0 = FALSE 1 = TRUE

23.3.19.3 MBDRC_CONFIG_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x0030de51 (0bxxxxxxx00011000011011110010 1xx01)

Bit	Reset	Description
24:16	0x30	RMS_OFFSET: RMS Offset used during RMS detection(-16dB to +15.9375dB) (Q5.4 format)
15	UNBIAS	BIAS_UNBIAS: Rounding Option across all Rounding Operations of MBDRC Biased = Round to plus infinity (both positive and negative numbers) 1 = UNBIAS: Unbias = Round to plus infinity (positive) 0 = BIAS
14	PEAK	PEAK_RMS: Mode Select for Peak Detection in SideChain 0 = RMS 1 = PEAK

Bit	Reset	Description
13	ALL_PASS_TREE	FILTER_STRUC: Selects between a custom Butterfly Structure with fixed BiQuad stages (ALL_PASS_TREE) 0 = ALL_PASS_TREE: or a programmable structure where per band biquads can be programmed (FLEX) 1 = FLEX
12:8	0x1e	SHIFT_CTRL: Shift Control for each BiQuad Stage - this depends on the format of coefficient programmed Default to 30 - current coefficients are all with 2.30 format. Software can only change it when the format of coefficients is changed. Otherwise, always keep 30.
7:4	N32	FRAME_SIZE: Number of Samples per "frame" that will be used to compute the sidechain calculation 0 = N1: Has to be a power of 2 1 = N2: Maximum Frame size that hardware can support is 64 (FRAME_SIZE=6) 2 = N4: It will result in incorrect output if it is set bigger than 6 3 = N8 4 = N16 5 = N32 6 = N64
1:0	FULLBAND	MBDRC_MODE: MBDRC Mode of Operation Filtering - LowPass, BandPass and HighPass, SideChain, and Gain Application for the three bands 0 = BYPASS: Bypass - No Filtering, No Side Chain, No Gain Application (DISABLE) 1 = FULLBAND: Full Band - No Filtering, But SideChain and Gain Application - for a single Band 2 = DUALBAND: Filtering - LowPass and HighPass, SideChain and Gain Application for the two Band 3 = MULTIBAND

23.3.19.4 MBDRC_CHAN_MASK_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000011)

Bit	Reset	Description
7:0	0x3	CHAN_MASK_EN: An N bit field - a "1" on field, indicates perform DRC on that channel a "0" on field, indicates DRC will not be done on that channel, i.e., no SideChain Calculation, and Unity Gain will be applied Bit(N-1)-0 = Masks for Channels(N-1)-0

23.3.19.5 MBDRC_MASTER_VOLUME_0

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MASTER_VOLUME: Feedback of Volume Control to MBDRC (if Volume Control is placed before MBDRC) (Signed Q9.23 format) (Default is 0 dB)

23.3.19.6 MBDRC_FAST_FACTOR_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x30000800 (0b00110000000000000000100000000000)

Bit	Reset	Description
31:16	0x3000	FR_FACTOR: Gain Smoothing: FastRelease is applied if CurrentGain/PreviousGain < FRFactor default: 12288 (0.375) (Q1.15 format) positive value only. Defining them as negative values will result in incorrect results.
15:0	0x800	FA_FACTOR: Gain Smoothing: FastAttack is applied if PreviousLevel/CurrentLevel < FAFactor default: 2048 (0.0625) (Q1.15 format) positive value only. Defining them as negative values will result in incorrect results

23.3.19.7 MBDRC_IIR_CONFIG_0

The following registers that are indexed by the MAX_BANDS (0,1,2) correspond to low/mid/high as follows:

- Band_0 = Low Band
- Band_1 = Mid Band
- Band_2 = High Band

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x38..0x43 | Read/Write: R/W | Reset: 0x00000005 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0101)

Bit	Reset	Description
3:0	0x5	NUM_STAGES: Number of Stages - Maximum of 6,8,6 - only applicable in ALL_PALL_TREE mode - Here number is actually N-1, for max 6 - this number should be 5

23.3.19.8 MBDRC_INATTACK_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x44..0x4f | Read/Write: R/W | Reset: 0x3e48590c (0b00111110010010000101100100001100)

Bit	Reset	Description
31:0	0x3e48590c	IN_ATTACK_TC: Attack Time Constant for Input envelope detection (different for each band) (1 - exp((-1.0 * BufferSize) / (Fs * INattackTC))) default: 1044928780 (0.001sec @ 48KHz with 32 BufferSize) positive value only. Defining them as negative values will result in incorrect results

23.3.19.9 MBDRC_INRELEASE_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x50..0x5b | Read/Write: R/W | Reset: 0x08414e9f (0b00001000010000010100111010011111)

Bit	Reset	Description
31:0	0x8414e9f	IN_RELEASE_TC:Release Time Constant for Input envelope detection (different for each band)(1 - exp((-1.0 * BufferSize) / (Fs * INrttackTC)))default: 138497695 (0.01sec @ 48KHz with 32 BufferSize) positive value only. Defining them as negative values will result in incorrect results

23.3.19.10 MBDRC_FASTATTACK_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x5c..0x67 | Read/Write: R/W | Reset: 0x7fffffff (0b01111111111111111111111111111111)

Bit	Reset	Description
31:0	0x7fffffff	FAST_ATTACK_TC:Fast Attack Time Constant for Input envelope detection (different for each band)(1 - exp((-1.0 * BufferSize) / (Fs * FastAttackTC)))default: 2147483647 (0.00001sec @ 48KHz with 32 BufferSize) positive value only. Defining them as negative values will result in incorrect results

23.3.19.11 MBDRC_IN_THRESH_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x68..0x73 | Read/Write: R/W | Reset: 0x06145082 (0b00000110000101000101000010000010)

Bit	Reset	Description
31:24	0x6	IN_THRESH_4TH:(default: 6) Fourth Knee for the 5 gain segments, which is close to 0 dB (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)
23:16	0x14	IN_THRESH_3RD:(default: 20) Third Knee for the 5 gain segments (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)
15:8	0x50	IN_THRESH_2ND:(default: 80) Second Knee for the 5 gain segments (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)
7:0	0x82	IN_THRESH_1ST:(default: 130) First Knee for the 5 gain segments, which is close to -127.5 dB (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)

23.3.19.12 MBDRC_OUT_THRESH_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x74..0x7f | Read/Write: R/W | Reset: 0x060d379b (0b00000110000011010011011110011011)

Bit	Reset	Description
31:24	0x6	OUT_THRESH_4TH:(default: 6) Fourth output point corresponding to Fourth IN_THRESH, (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)
23:16	0xd	OUT_THRESH_3RD:(default: 13) Third output point corresponding to Third IN_THRESH, (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)
15:8	0x37	OUT_THRESH_2ND:(default: 55) Second output point corresponding to Second IN_THRESH, (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)
7:0	0x9b	OUT_THRESH_1ST:(default: 155) First output point corresponding to First IN_THRESH, (0dB to -127.5dB in 0.5dB step size) (After conversion Thresh -40 is provided as 40)

23.3.19.13 MBDRC_RATIO_1ST_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x80..0x8b | Read/Write: R/W | Reset: 0x0000a000 (0bxxxxxxxxxxxxxxxx1010000000000000)

Bit	Reset	Description
15:0	0xa000	RATIO_1ST:(default: 40960) Output to Input ratio for first gain segment, which is close to -127.5 dB. (Q4.12 format)

23.3.19.14 MBDRC_RATIO_2ND_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x8c..0x97 | Read/Write: R/W | Reset: 0x00002000 (0bxxxxxxxxxxxxxxxx0010000000000000)

Bit	Reset	Description
15:0	0x2000	RATIO_2ND:(default: 8192) Output to Input ratio for second gain segment. (Q4.12 format)

23.3.19.15 MBDRC_RATIO_3RD_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x98..0xa3 | Read/Write: R/W | Reset: 0x00000b33 (0bxxxxxxxxxxxxxxxx0000101100110011)

Bit	Reset	Description
15:0	0xb33	RATIO_3RD:(default: 2867) Output to Input ratio for third gain segment. (Q4.12 format)

23.3.19.16 MBDRC_RATIO_4TH_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xa4..0xaf | Read/Write: R/W | Reset: 0x00000800 (0bxxxxxxxxxxxxxxxx0000100000000000)

Bit	Reset	Description
15:0	0x800	RATIO_4TH:(default: 2048) Output to Input ratio for fourth gain segment. (Q4.12 format)

23.3.19.17 MBDRC_RATIO_5TH_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xb0..0xbb | Read/Write: R/W | Reset: 0x0000019a (0bxxxxxxxxxxxxxxxx0000000110011010)

Bit	Reset	Description
15:0	0x19a	RATIO_5TH:(default: 410) Output to Input ratio for fifth gain segment, which is close to 0 dB. (Q4.12 format)

23.3.19.18 MBDRC_MAKEUP_GAIN_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xbc..0xc7 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000010)

Bit	Reset	Description
5:0	0x2	MAKEUP_GAIN: Makeup Gain (5.1 Format, -16dB to 15.5dB in 0.5dB step size)

23.3.19.19 MBDRC_INITGAIN_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xc8..0xd3 | Read/Write: R/W | Reset: 0x00066666 (0b00000000000001100110011001100110)

Bit	Reset	Description
31:0	0x66666	GAIN_INIT:Starting value for the gain to be applied in linear (Gain range in dB = -127dB to 54dB)default: 419430 (0.1 i.e., -20dB) (Q10.22 format)

23.3.19.20 MBDRC_GAINATTACK_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xd4..0xdf | Read/Write: R/W | Reset: 0x00d9ba0e (0b00000000110110011011101000001110)

Bit	Reset	Description
31:0	0xd9ba0e	GAIN_ATTACK_TC: Attack Time Constant for gain smoothing (different for each band)(1 - exp((-1.0 * BufferSize) / (Fs * GAINattackTC)))default: 14268942 (0.1sec @ 48KHz with 32 BufferSize) (Q1.31 format) positive value only. Defining them as negative values will result in incorrect results

23.3.19.21 MBDRC_GAINRELEASE_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xe0..0xeb | Read/Write: R/W | Reset: 0x3e48590c (0b00111110010010000101100100001100)

Bit	Reset	Description
31:0	0x3e48590c	GAIN_RELEASE_TC:Release Time Constant for gain smoothing (different for each band)(1 - exp((-1.0 * BufferSize) / (Fs * GAINreleaseTC))) default: 1044928780 (0.001sec @ 48KHz with 32 BufferSize) (Q1.31 format) positive value only. Defining them as negative values will result in incorrect results

23.3.19.22 MBDRC_FASTRELEASE_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xec..0xf7 | Read/Write: R/W | Reset: 0x7ffff26a (0b01111111111111111111001001101010)

Bit	Reset	Description
31:0	0x7ffff26a	FAST_RELEASE_TC: Fast Release Time Constant for gain smoothing (different for each band)(1 - exp((-1.0 * BufferSize) / (Fs * FastRelease)))default: 2147480170 (0.00005sec @ 48KHz with 32 BufferSize) (Q1.31 format) positive value only. Defining them as negative values will result in incorrect results

23.3.19.23 MBDRC_AHUBRAMCTL_MBDRC_CTRL_0

COEFFICIENT RAM control for all bands

1. There is one Coefficient RAM for each Band
   - FULLBAND -> No need to program the coefficient RAM, since there is no Biquad Filtering
   - DUALBAND -> First two bands' coefficient RAM need to be programmed.

- MULTIBAND -> Three bands' coefficient RAMs needs to be programmed.

2. The coefficients inside each RAM are organized in the following order:
   - 'd00: Biquad1-b0
   - 'd01: Biquad1-b1
   - 'd02: Biquad1-b2 > Biquad 1
   - 'd03: Biquad1-a1
   - 'd04: Biquad1-a2
   - 'd05: Biquad2-b0
   - 'd06: Biquad2-b1
   - 'd07: Biquad2-b2 > Biquad 2
   - 'd08: Biquad2-a1
   - 'd09: Biquad2-a2
   - 'd35: Biquad8-b0
   - 'd36: Biquad8-b1
   - 'd37: Biquad8-b2 > Biquad 8
   - 'd38: Biquad8-a1
   - 'd39: Biquad8-a2
3. The order of coefficients is b0,b1,b2,a1,a2 (5 set)
4. y(n) = b0*x(n) + s1(n)

   s1(n) = b1*x(n) - a1*y(n) + s2(n)

   s2(n) = b2*x(n) - a2*y(n)
5. For MULTIBAND mode in the ALL_PASS_TREE structure, for the third band, the coefficient starting address should be 'd15 (the fourth biquad), since hardware only maintains one read pointer which is used for all the coefficient RAMs
6. Current format is 2.30. If the format changed, the SHIFT_CTRL value should be updated correspondingly.

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0xf8..0x103 | Read/Write: R/W | Reset: 0xX0004000 (0bxxxxxxxx00000000x100xxx000000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
23:16	RW	0x0	SEQ_READ_COUNT
14	RW	WRITE	RW: 0 = READ 1 = WRITE
13	RW	DISABLE	ADDR_INIT_EN: 0 = DISABLE 1 = ENABLE
12	RW	DISABLE	SEQ_ACCESS_EN: 0 = DISABLE 1 = ENABLE
8:0	RW	0x0	RAM_ADDR

23.3.19.24 MBDRC_AHUBRAMCTL_MBDRC_DATA_0

This is an array of 3 identical register entries; the register fields below apply to each entry.

Offset: 0x104..0x10f | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

23.3.20 ADSP Peripheral Registers

23.3.20.1 ADSP_PERIPH_CPUIF_ICCICR_0

Offset: 0x100 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCICR: CPU Interface Control Register

23.3.20.2 ADSP_PERIPH_CPUIF_ICCPMR_0

Offset: 0x104 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCPMR: Interrupt Priority Mask Register

23.3.20.3 ADSP_PERIPH_CPUIF_ICCBPR_0

Offset: 0x108 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCBPR: Binary Point Register

23.3.20.4 ADSP_PERIPH_CPUIF_ICCIAR_0

Offset: 0x10c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCIAR: Interrupt Acknowledge Register

23.3.20.5 ADSP_PERIPH_CPUIF_ICCEOIR_0

Offset: 0x110 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCEOIR: End of Interrupt Register

23.3.20.6 ADSP_PERIPH_CPUIF_ICCRPR_0

Offset: 0x114 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCRPR: Running Priority Register

23.3.20.7 ADSP_PERIPH_CPUIF_ICCHPPIR_0

Offset: 0x118 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCHPPIR: Highest Priority Pending Interrupt Register

23.3.20.8 ADSP_PERIPH_CPUIF_ICCABPR_0

Offset: 0x11c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GICC_ABPR: Aliased Binary Point Register

23.3.20.9 ADSP_PERIPH_CPUIF_ICCIDR_0

Offset: 0x1fc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCIDR: CPU Interface Implementer Identification Register

23.3.20.10 ADSP_PERIPH_DISTRIBUTOR_ICDDCR_0

Offset: 0x1000 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDDCR: Distributor Control Register

23.3.20.11 ADSP_PERIPH_DISTRIBUTOR_ICDICTR_0

Offset: 0x1004 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDICTR: Interrupt Controller Type Register

23.3.20.12 ADSP_PERIPH_DISTRIBUTOR_ICDIIDR_0

Offset: 0x1008 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDIIDR: Distributor Implementer Identification Register

23.3.20.13 ADSP_PERIPH_DISTRIBUTOR_ICDISR<n>_0

There are 8 Interrupt Group registers, where n = 0 through 7.

Offset: 0x1080 + (n * 0x04) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDISR<n>: Interrupt Group Registers

23.3.20.14 ADSP_PERIPH_DISTRIBUTOR_ICDISER<n>_0

There are 8 Interrupt Set Enable registers, where n = 0 through 7.

Offset: 0x1100 + (n * 0x04) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	**X**	**ICDISER<n>: Interrupt Set-Enable Registers**

23.3.20.15 ADSP_PERIPH_DISTRIBUTOR_ICDICER<n>_0

There are 8 Interrupt Clear Enable registers, where n = 0 through 7.

Offset: 0x1180 + (n * 0x04) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDICER<n>: Interrupt Clear-Enable Registers

23.3.20.16 ADSP_PERIPH_DISTRIBUTOR_ICDISPR<n>_0

There are 8 Interrupt Set Pending registers, where n = 0 through 7.

Offset: 0x1200 + (n * 0x04) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDISPR<n>: Interrupt Set-Pending Registers

23.3.20.17 ADSP_PERIPH_DISTRIBUTOR_ICDICPR<n>_0

There are 8 Interrupt Clear Pending registers, where n = 0 through 7.

Offset: 0x1280 + (n * 0x04) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDICPR<n>: Interrupt Clear-Pending Registers

23.3.20.18 ADSP_PERIPH_DISTRIBUTOR_ICDABR<n>_0

There are 8 Interrupt Active Bit registers, where n = 0 through 7.

Offset: 0x1300 + (n * 0x04) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDABR<n>: Interrupt Active Bit Registers

23.3.20.19 ADSP_PERIPH_DISTRIBUTOR_ICDIPR<n>_0

There are 64 Interrupt Priority registers, where n = 0 through 63.

Offset: 0x1400 + (n * 0x04) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDIPR<n>: Interrupt Priority Registers

23.3.20.20 ADSP_PERIPH_DISTRIBUTOR_ICDIPTR<n>_0

There are 64 Interrupt Processor Targets registers, where n = 0 through 63.

Offset: 0x1800 + (n * 0x04) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDIPTR<n>: Interrupt Processor Targets Registers

23.3.20.21 ADSP_PERIPH_DISTRIBUTOR_ICDICFR<n>_0

There are 16 Interrupt Configuration registers, where n = 0 through 15.

Offset: 0x1c00 + (n * 0x04) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDICFR<n>: Interrupt Configuration Registers

23.3.20.22 ADSP_PERIPH_DISTRIBUTOR_ICPPISR_0

Offset: 0x1d00 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPPISR: Private Peripheral Interrupt Status Register

23.3.20.23 ADSP_PERIPH_DISTRIBUTOR_ICSPISR<n>_0

There are 7 Shared Peripheral Interrupt Status registers, where n = 0 through 6.

Offset: 0x1d04 + (n * 0x04) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICSPISR<n>: Shared Peripheral Interrupt Status Registers

23.3.20.24 ADSP_PERIPH_DISTRIBUTOR_ICDSGIR_0

Offset: 0x1f00 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICDSGIR: Software Generated Interrupt Register

23.3.20.25 ADSP_PERIPH_DISTRIBUTOR_ICPIDR0_0

Offset: 0x1fd0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR0: Peripheral ID4 Register

23.3.20.26 ADSP_PERIPH_DISTRIBUTOR_ICPIDR1_0

Offset: 0x1fd4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR1: Peripheral ID5 Register

23.3.20.27 ADSP_PERIPH_DISTRIBUTOR_ICPIDR2_0

Offset: 0x1fd8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR2: Peripheral ID6 Register

23.3.20.28 ADSP_PERIPH_DISTRIBUTOR_ICPIDR3_0

Offset: 0x1fdc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR3: Peripheral ID7 Register

23.3.20.29 ADSP_PERIPH_DISTRIBUTOR_ICPIDR4_0

Offset: 0x1fe0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR4: Peripheral ID0 Register

23.3.20.30 ADSP_PERIPH_DISTRIBUTOR_ICPIDR5_0

Offset: 0x1fe4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR5: Peripheral ID1 Register

23.3.20.31 ADSP_PERIPH_DISTRIBUTOR_ICPIDR6_0

Offset: 0x1fe8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR6: Peripheral ID2 Register

23.3.20.32 ADSP_PERIPH_DISTRIBUTOR_ICPIDR7_0

Offset: 0x1fec | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICPIDR7: Peripheral ID3 Register

23.3.20.33 ADSP_PERIPH_DISTRIBUTOR_ICCIDR0_0

There are 4 Component ID registers, where n = 0 through 3.

Offset: 0x1ff0 + (n * 0x04) | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ICCIDR<n>: Component ID<n> Register

23.3.21 Audio Miscellaneous Registers

23.3.21.1 AMISC_AMISC_CONFIG_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	DISABLE_ERROR_RESPONSE: Do not send error on AXI bus on an invalid access

23.3.21.2 AMISC_ADSP_CONFIG_0

Offset: 0x4 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xe0800000 (0b111xx0001x0000xxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	0x7	MAXCLKLATENCY:
26:23	0x1	CLUSTERID
21	0x0	VINITHI
20	0x0	CFGSDISABLE
19	0x0	CP15SDISABLE
18	0x0	EVENTI

23.3.21.3 AMISC_ADSP_PERIPHBASE_0

Offset: 0x8 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00c00000 (0b00000000110000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:13	0x600	PERIPHBASE

23.3.21.4 AMISC_ADSP_L2_CONFIG_0

Offset: 0xc | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x02000000 (0bx000001xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:25	0x1	CACHEID

23.3.21.5 AMISC_ADSP_L2_REGFILEBASE_0

Offset: 0x10 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00c02000 (0b00000000110000000010xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0xc02	REGFILEBASE

23.3.21.6 AMISC_ADSP_STATUS_0

Offset: 0x14 | Read/Write: RO | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	L2_IDLE
30	X	L2_CLKSTOPPED
29	X	DBGNOPWRDWN

23.3.21.7 AMISC_ADSP_AGIC_CONFIG_0

Offset: 0x18 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CFGSDISABLE

23.3.21.8 AMISC_SHRD_SMP_STA_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SHRD_SMP_STA: Current value of the 32 semaphores

23.3.21.9 AMISC_SHRD_SMP_STA_SET_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SHRD_SMP_STA_SET: Writing bit i to 1 sets the corresponding semaphore bit.

23.3.21.10 AMISC_SHRD_SMP_STA_CLR_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SHRD_SMP_STA_CLR: Writing bit i to 1 clears corresponding semaphore bit.

23.3.21.11 AMISC_CPU_ARBGNT_STATUS_0

Arbitration semaphores provide a mechanism by which the two processors can arbitrate for the use of various resources. These semaphores provide a hardware locking mechanism, so that when a processor is already using a resource, the second processor is not granted that resource. There are 32 bits of Arbitration semaphores provided in the system. The hardware does not enforce any resource association to these bits. It is left to the firmware to assign and use these bits.

The setup/usage of the Arbitration Semaphores is described in Chapter 4: Semaphores.

The Arbitration Semaphores can also generate an interrupt when a hardware resource becomes available. The registers in this module configure these interrupts. When a 1 is set in the corresponding bit position of the Arbitration Semaphore Interrupt Source Register (CPU_enable or DSP_enable), an interrupt will be generated when the processor achieves Grant Status for that resource.

The current Grant status can be viewed in the CPU_STATUS or DSP_STATUS registers.

Semaphore Granted Status Register

Offset: 0x28 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	ARBGNT_STATUS: A one in any bit indicates that the processor reading this register as granted status for that bit. A zero indicates semaphore not granted.

23.3.21.12 AMISC_CPU_ARBGNT_GET_0

Request Arbitration Semaphore Register

Offset: 0x2c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	ARBGNT_GET: Writing a one in any bit is a request for that semaphore bit by the processor performing the register write.

23.3.21.13 AMISC_CPU_ARBGNT_PUT_0

Arbitration Semaphore Put Request Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	ARBGNT_PUT: Writing a one in any bit will clear the corresponding semaphore bit by the processor performing the register write.

23.3.21.14 AMISC_ADSP_ARBGNT_STATUS_0

Semaphore Granted Status Register

Offset: 0x34 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	ARBGNT_STATUS: A one in any bit indicates that the processor reading this register as granted status for that bit. A zero indicates semaphore not granted.

23.3.21.15 AMISC_ADSP_ARBGNT_GET_0

Request Arbitration Semaphore Register

Offset: 0x38 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	ARBGNT_GET: Writing a one in any bit is a request for that semaphore bit by the processor performing the register write.

23.3.21.16 AMISC_ADSP_ARBGNT_PUT_0

Arbitration Semaphore Put Request Register

Offset: 0x3c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	ARBGNT_PUT: Writing a one in any bit will clear the corresponding semaphore bit by the processor performing the register write.

23.3.21.17 AMISC_ADSP_ARBGNT_REQ_STATUS_0

Arbitration Request Pending Status (1=PENDING) Register

Offset: 0x40 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	REQ_STATUS: A one in any bit indicates a request pending status from the AMISC_ADSP_ARBGNT_GET_0 register. The corresponding bits are set when the request for the individual resource is pending.

23.3.21.18 AMISC_CPU_ARBGNT_REQ_STATUS_0

Arbitration Request Pending Status (1=PENDING) Register

Offset: 0x44 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	REQ_STATUS: REQ_STATUS: A one in any bit indicates a request pending status from the AMISC_CPU_ARBGNT_GET_0 register. The corresponding bits are set when the request for the individual resource is pending.

23.3.21.19 AMISC_TSC_0

The TSC register contains the MSB 32 bits of the TSC signal on the first read. On the 2nd read, it contains the bottom 32 bits of the TSC signal. The TSC signal is zero-extended from 56 bits to 64 bits.

Offset: 0x48 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TSC

23.3.21.20 AMISC_AMISC_DEBUG_0

Offset: 0x4c | Read/Write: RO | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TSC_SET: If 1, indicates that lower 32 bits will be read from TSC register. If 0, indicates that upper 32 bits will be read from the TSC register.

23.3.21.21 AMISC_IDLE_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x80000000 (0b1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x1	IDLE_EN

23.3.21.22 AMISC_ACTMON_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x7f800000 (0b011111111xxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CNT_ENABLE: Enable the ADSP side counter
30:23	0xff	CNT_TARGET: Count target minus 1 before signal to the ACTMON. The ACTMON will be signalled when the counter = CNT_TARGET and another ~standbywfi cycle is detected (i.e., CNT_TARGET+1 events).

23.3.21.23 AMISC_SHRD_MBOX_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x58..0x67 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	TAG
30:0	0x0	DATA

23.3.21.24 AMISC_SPARE_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SPARE

CHAPTER 24: DISPLAY CONTROLLER

The Tegra® X1 architecture has two entirely independent display controllers. They can support two independent display devices, typically a local display panel and an external HDMI TV or DP monitor. Other configurations are possible such as two local panels. Each display controller can run at a different clock rate and drive a different resolution panel.

Refer to Chapter 27: Display Interfaces: HDMI and DisplayPort for information on DisplayPort, eDP, HDMI, and HDCP. Refer to the “Color Decompression Engine” section for information on CDE.

24.1 Features

Key features include:

- Serial output resource (SOR) supports DisplayPort (DP) or eDP concurrently with HDMI or MIPI-DSI
- Three full function display windows, A, B and C, supporting scaling and blending
- Two additional windows (collectively known as “Simple Windows”):
  - Window D: Fourth general-purpose window
  - Window T: TrustZone® Secure display
- Color Decompression Engine (CDE) support
- Display Stream Compression (DSC) support
- Enhanced PRISM2 display with run per frame on VPULSE2 and smooth K0
- Maximum panel resolution:
  - Internal panel
    - DSI

    Dual link 3200 x 2000 @ 60Hz (2D – portrait/landscape)

    Single link 2560 x 1440 @ 60Hz (2D – portrait/landscape)
    - eDP

    3840 x 2160 @ 60 Hz (2D – portrait/landscape)
  - External panel
    - HDMI 4096 x 2160 @ 60 Hz (2D – landscape)
    - HDMI 1920 x 1080 @ 60 Hz (3D – frame packed mode)
- Use of line buffering to avoid re-fetching data from memory on subsequent lines
- Line buffer free dithering: temporal dithering
- HDMI 1080i and some HDMI YUV output formats are supported (BT.601, BT.709)
- Efficient fetch for scaling (select windows)
- Improved handling of display buffer underflow (select windows)
- Supports high-quality scaling/filtering (select windows)
- Supports up to Quad Full HD resolution (3840 x 2160)
- Support for Image Rotation in hardware
- Supports semi-planar YUV (input surface format)

- Extending the cursor size to 256x256 with full color and alpha-blending
- Display Color Management unit (based on a LUT and CSC in the output path)
- Layered blending among all windows
- Independent cursor update/activate control

24.1.1 Output Capabilities

- 1 or 2 internal panels
  - MIPI-DSI on two 4-lane DSI channels, operating either independently for one or two panels, or ganged together as 2x4 for a single panel
    - Maximum panel size of 3840 x 1920 @ 60 Hz using 2x4-lane ganged DSI channels
    - Independent resolution and pixel clock when not ganged
  - eDP supports 18/24 bpp, 1/2/4 data lanes, RBR/HBR/HBR2 up to 3840x2160 @ 60Hz
  - DSC supported by MIPI-DSI
    - Maximum panel size of 4096 x 2304 @ 60 Hz
  - 1 external display
  - On HDMI. Maximum 1920x1080p @ 60 Hz (3D) or 4096x2160 @ 60Hz (2D)
  - DisplayPort or HDMI 4096x2160 @ 60 Hz (2D)
  - With HDCP and multi-channel audio (HDMI 2.0 and DHCP 2.2)

Note: *The display controller only outputs RGB444 or YUV444 format to the output interface controllers (DSI, SOR). Refer to the chip output formats in the individual interface sections.*

24.1.2 Color Management Unit (CMU)

- Gamma (tone curve) conversion
- Color gamut conversion
- Allow conversion of sRGB content for a non-sRGB panel, for predictable colors (color matching)
- Enhanced PRISM2 display for these panels, as the PRISM2 algorithm assumes sRGB gamma
- Enable TCON and panels with non sRGB response
- Allows color calibrated displays, and enables end-to-end Camera / ISP -> Display color quality

24.1.3 Windows A/B/C

- Data Formats include (see Section 24.3.5: Surface Formats for a complete list):
  - B4G4R4A4
  - B5G5R5A, AB5G5R5, B5G6R5
  - B8G8R8A8, R8G8B8A8
  - YCbCr420P, YCbCr422P, YCbCr422RP (planar format)
  - YCbCr422 (packed format)
  - YCrCb420SP, YCrCb422SP (semiplanar format)
  - Palletized format limited to 8-bit mode
- Three 256x8-bit RAMs (color palette) used for gamma correction

- Horizontal and Vertical Flip
- Horizontal and Vertical Scaling
  - Horizontal scale-down by up to 4x, subject to clock speed limitations
  - Vertical scale-down by up to 4x for 2 bytes per pixel or 2x for 4 bytes per pixel, again subject to clock speed limitations
  - Scale-up by up to 4000x (from 1 pixel to 4000 pixels)
- Filtering (Windows A/B/C)
  - 6-tap, 16-phase programmable H filter
  - 2-tap, 16-phase programmable V filter
- YCbCr to RGB color space converter (Windows A/B/C)
- Digital Vibrance (8-level)

The table below describes a breakdown of features for a Tegra X1 Display on a window basis. Note that rotation is supported up to 2560x1600. Also note that system bandwidth, power, and other constraints might limit what combinations of resolution and features are achievable.

Table 120: Window Feature Summary

Head	Maximum Window Resolution @ 60 fps	Window	Planar Rotation	Packed Rotation	Compressed Surfaces (32 bpp RGBA)
A Without Rotation	4096x4096	A, B, C	-	-	Y
	4096x4096	D, T	-	-	N
	256x256	Cursor	-	-	N
A With Rotation	4096x2304	A	Y	Y	Y
	4096x4096		N	Y	Y
	4096x4096	B	N	Y	Y
		C	N	Y	Y
B Without Rotation	4096x4096	A	-	-	Y
		B	-	-	Y
		C	-	-	Y
	256x256	Cursor	-	-	N

24.1.4 Windows D/T (Simple Windows)

Windows D and T are similar, generic windows with limited features for supporting status bars, soft buttons, and similar use cases. Window T has additional TrustZone semantics and limitations.

Windows D/T, collectively known as “simple windows”, support a subset of the registers used by the full-featured windows A/B/C.

The following table summarizes the differences among the two windows.

Table 121: Windows D/T Summary

Feature	Window D	Window T
32/16 bpp RGBA	Y	Y
Syncpt	Y	N
STATE_CONTROL shadow	Y	N
TrustZone Secure mode	N	Y
Indirect access (DISPLAY_WINDOW_HEADER)	Y	N

Table 121: Windows D/T Summary

Feature	Window D	Window T
Class (Host1x command) access	Y	N [secure mode]

24.1.4.1 Feature Limitations

Windows D/T have similar window datapath features. Unlike window T, window D has syncpt increment and state control similar to windows A/B/C. Limitations of windows D/T relative to A/B/C include:

- Color depth is limited to a subset of RGB/RGBA single-plane formats. YCbCr/YUV/palette formats are not supported. The following formats are supported:
  - T_R8G8B8A8 – Tegra 4-style RGBA cursor uses
  - T_A8R8G8B8 – Tegra 3/Tegra 4 common 32-bit format
  - T_A8B8G8R8 – Tegra 3/Tegra 4 common 32-bit format
  - T_R5G6B5 – Best quality 16bpp format
  - T_A4R4G4B4 – 16bpp format with 4-bit alpha
  - T_A1R5G5B5 – 16bpp with 1-bit alpha format, better quality
- No scaling or filtering (DDA_INCREMENT, H/V_FILTER_ENABLE)
- Blending is limited to per-pixel alpha (ALPHA_WEIGHT and PREMULT_WEIGHT) and fixed alpha (FIX_WEIGHT). No color key or dependent weight (i.e., no dependence on non-secure pixels)
- The following features are not supported:
  - Block linear
  - H_DIRECTION, V_DIRECTION
  - Rotation (SCAN_COLUMN)
  - DV (digital vibrance)
  - BYTE_SWAP
  - TILE_MODE
  - H/V_INITIAL_DDA
  - DDA_INCREMENT (no scaling)

Window T has some additional feature limitations:

- No syncpt increment
- No indirect access using DISPLAY_WINDOW_HEADER – only window direct space is supported
- No class writes in secure mode. Class writes are always non-secure and therefore cannot be used in secure mode.
- Register activation and shadow control do not use STATE_CONTROL/STATE_ACCESS/REG_ACT_CONTROL. Separate registers are used.

24.1.4.2 Blending

Window D/T supports a subset of window blending registers. Because color key is not supported, only MATCH case exists.

24.1.4.3 Window T TrustZone Support

The display controller supports the ability to display TrustZone secured content in window T. Registers and memory interface in window T are separate from the other windows.

The display controller can be configured at boot time between Secure and non-Secure modes. In Secure mode, window T is composited on top of 4 non-secure windows, and the cursor is disabled. The Non-secure OS can continue to run as normal, but

its windows maybe obscured. All other semantics for the non-Secure OS are unchanged, including register reads, sync points, and interrupts.

TZ_SECURE is not intended for digital media DRM; the Video Protection Region is intended for that. The TZ_SECURE window T also does not have sufficient features for video use cases. TrustZone and VPR are orthogonal. TZ_SECURE does not require a VPR region, and VPR region access does not require TZ_SECURE.

Only displayA (internal display) has window T. But, both display A and displayB (external display) have TrustZone Secure host interfaces and implement the SECURE_CONTROL register. This allows the Secure OS to control the displayB output to DSI.

The display controller allows window T to use an alternate MC address-space ID (ASID) so that secure and non-secure address translation and attributes can be different. Window T is hardcoded to a different SWID (via the swid signal) from all other windows. Software can either program the two SWIDs to different ASIDs or to the same ASID.

The display controller allows disabling CRC in Secure mode, so that non-secure software cannot gain information about trusted display by examining the CRC.

Secure Mode Limitations

The TrustZone Secure OS mode has these limitations:

- No hardware cursor. The cursor could be used to spoof secure window.
- No use of display syncpt increment, interrupt or raise. The single interrupt line is shared by the whole head (all windows).
- DISPLAY_WINDOW_HEADER should not be used by TrustZone. TrustZone must use “window direct” address range only.
- Only direct Host1x writes can support TZ_SECURE accesses; indirect and CDMA/class writes might not.
- The Host1x channel should not be shared with non-TZ OS. If unavoidable, only Host1x direct read/write should be done. In particular, non-atomic Host1x sequences, such as indirect register writes, should be avoided.
- STATE_CONTROL, STATE_ACCESS, and REG_ACT_CONTROL must not be used by TrustZone. SECURE_STATE_CONTROL and WIN_T_STATE_ACCESS should be used instead.
- Display CRC includes the Secure Window which could confuse the non-Secure OS, if the latter uses CRC. This applies to functions like NvDPS.
- Secure window affects PRISM2 and thus backlight brightness. It might interact poorly with SD_WINDOW* region, for example.
- Bandwidth used by window T must be accounted for in DVFS, IMP, etc.

24.1.4.4 DVFS Support

Windows D/T are also responsible for generating DVFS readiness signals similar to those generated by MEMFETCH windows. The requirements for generating ready_for_latency_event for windows D/T DVFS are the same as for full featured windows.

24.2 Block Diagrams

Figure 83: Display Controller Front End Block Diagram

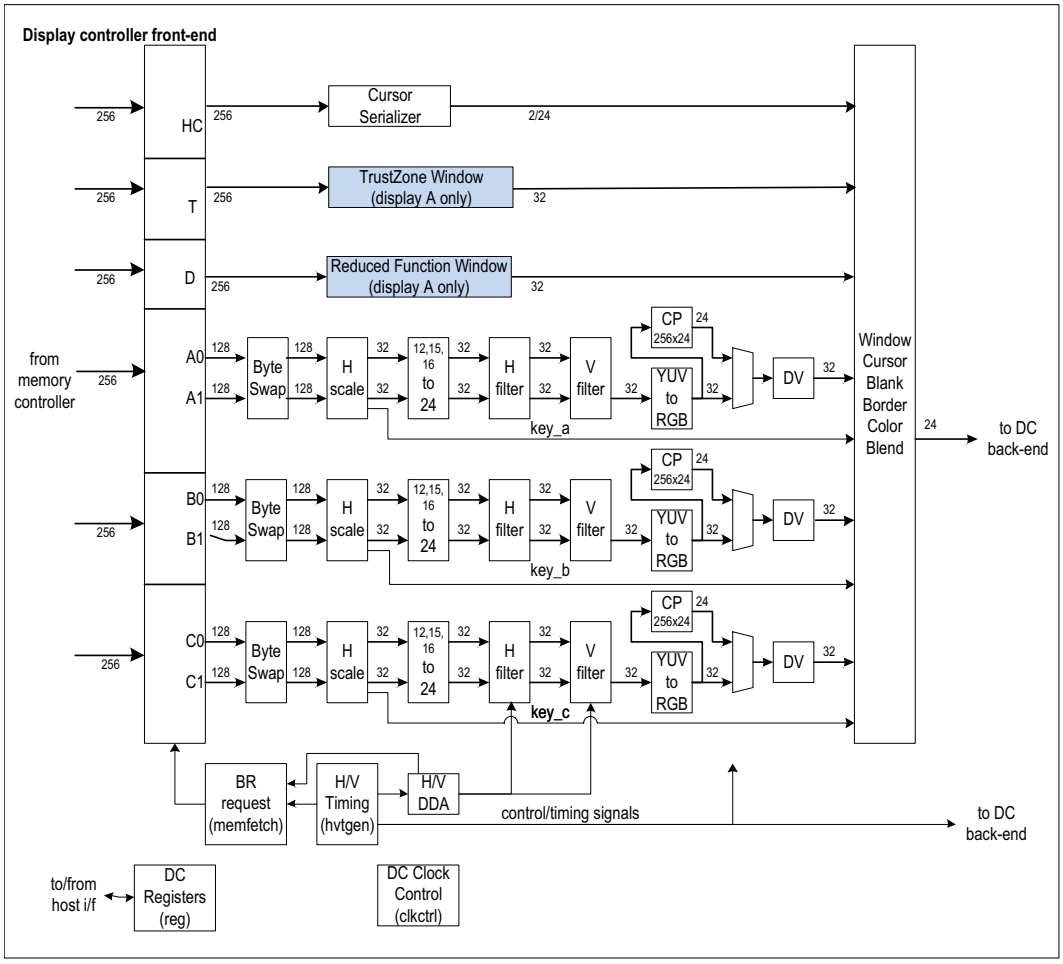
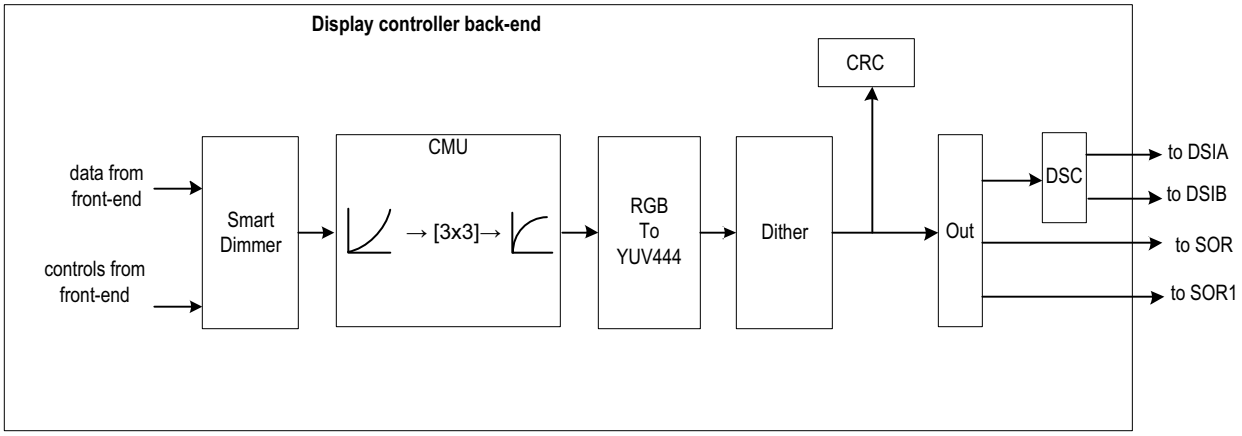


Figure 84: Display Controller Back End Block Diagram



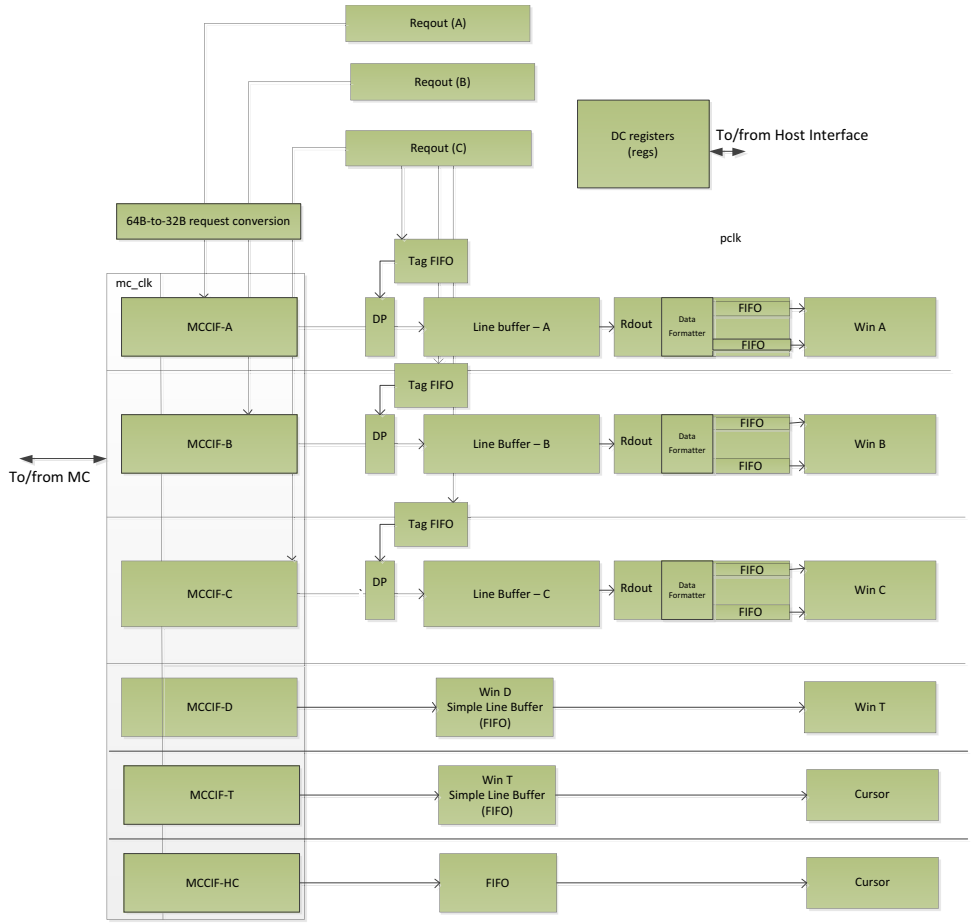
24.3 Display Controller Description

24.3.1 MEMFETCH

The following figure shows the top-level block diagram of the display memory interface for Tegra X1 devices. The diagram shows the memory fetch engines associated with the three full-featured windows and two simple windows in the display controller. Each window memory fetch engine has a line buffer and MCCIF. In addition to the windows, display also uses a memfetch for cursor.

Requests are issued to the memory and data is returned via MCCIF to the window line buffer. Data from the line buffer is subsequently read out, unpacked and passed to the window scaler and filter units. Windows D/T and the cursor fetch engines do not contain line buffers.

Figure 85: Memfetch Block Diagram



24.3.1.1 Request Engine

The request engine is responsible for fetching the window image from memory. Logic to translate the X, Y coordinates to a linear memory address is used to make requests to memory.

A pulse signal is sent to the request engine from the display timing generator at the beginning of the vertical blanking period to alert the request engine to start requesting the image data. Tagged requests are sent to the MCCIF. The thread ID specifying the line location of the return data is passed to the tag FIFO between the request and response engines. The thread ID is read by the data packer block when the return data is received and the data is steered to the appropriate line(s) in the line buffer.

The request engine must handle the following surface formats:

- Pitched linear surface

- Block linear surface
  - 16B x 2 lines – subset of 32B x 2 lines (rotation disabled)
  - 16B x 2 lines – subset of 16B x 4 lines (rotation enabled or CDE enabled)

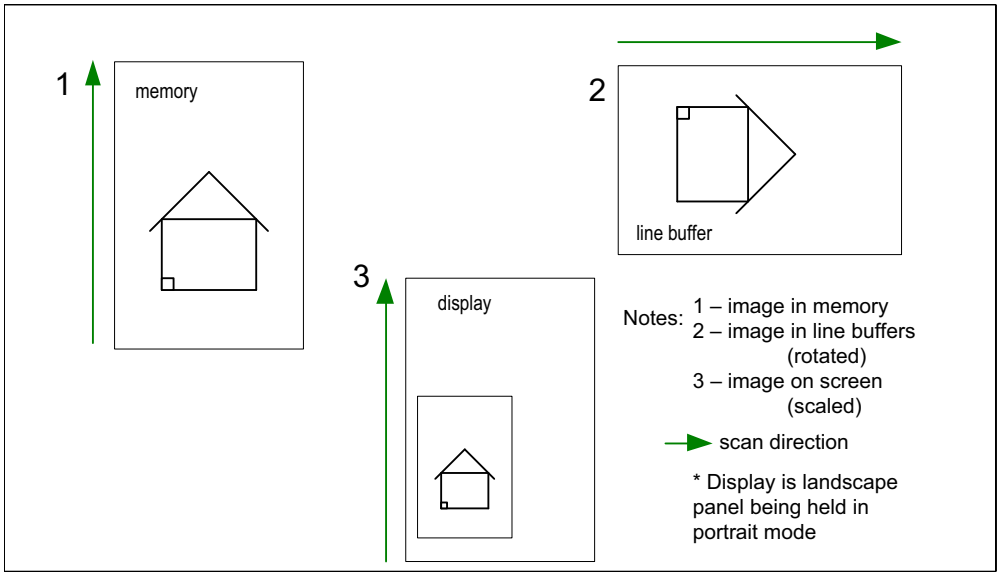
Requests are only stalled by:

- Backpressure from MCCIF
  - Occurs when MCCIF reorder buffer allocation is full
  - MCCIF reorder buffer size is 6KB
- Tag FIFO full
  - Each tag represents 64-byte line buffer entry
  - Current tag FIFO depth sized at 119 entries
  - Allows for slightly over 7KB requests
  - Tag FIFO depth should be slightly greater than possible MCCIF outstanding requests

Rotation Support

The display supports 0°, 90°□180°, or 270° rotation or image flip. To support rotation, the memfetch unit rotates the data at the input to the line buffers. That is, memfetch walks the Y axis and stores these columns in memory as rows to the display. The figure below illustrates the image orientation and shape at different stages in the pipeline for rotation.

Figure 86: Example Rotation Use Case



Memfetch only supports rotation in block linear mode. This is due to the amount of horizontal pixel data read in pitched linear versus block linear format. For pitched linear, 32B of horizontal data is fetched compared to block linear format where only 16B of horizontal data is fetched using the 16B x 4 line block linear mode. Since rotation switches rows and columns, the 4 horizontal pixels become 4 rows in the line buffer. Therefore the minimum number of lines when rotation is enabled is 4. The number of lines grows to 5 when scaling is enabled.

The request for 16Bx4 format must be communicated to the MCCIF via addressing bits 6 and 7. Memfetch will increment the Y instead of X coordinate for subsequent accesses and after reaching maximum Y will reset the coordinates to (X+16B, 0) for the next fetch of columns from memory. Selection of 32Bx2 or 16Bx4 is based on B_SCAN_COLUMN used to indicate rotation

The B_SCAN_COLUMN bit indicates if the image needs to be scanned out row by row or column by column. When this bit is enabled, the display needs to scan out lines column by column. This will provide the required 90°/270° rotated images. To achieve 90° or 270° rotation, software must use a combination of H_DIR, V_DIR, and SCAN_COLUMN as shown in the table

below. It illustrates how the image is scanned in to display by the memfetch unit for each combination of (SCAN_COLUMN, H_DIR, V_DIR).

Table 122: Rotation Cases

SCAN_COLUMN	H_DIR	V_DIR	Orientation
0	0	0	0 degrees
0	0	1	N/A – vertical flip
0	1	0	N/A – horizontal flip
0	1	1	180 degrees
1	0	0	N/A – vertical flip followed by rotation
1	0	1	90 degrees
1	1	0	270 degrees
1	1	1	N/A – horizontal flip followed by rotation

The window dimension registers can be confusing when rotation is introduced. Therefore it is important to note which registers apply to pre-rotation and which to post rotation.

The following registers apply to pre-rotation:

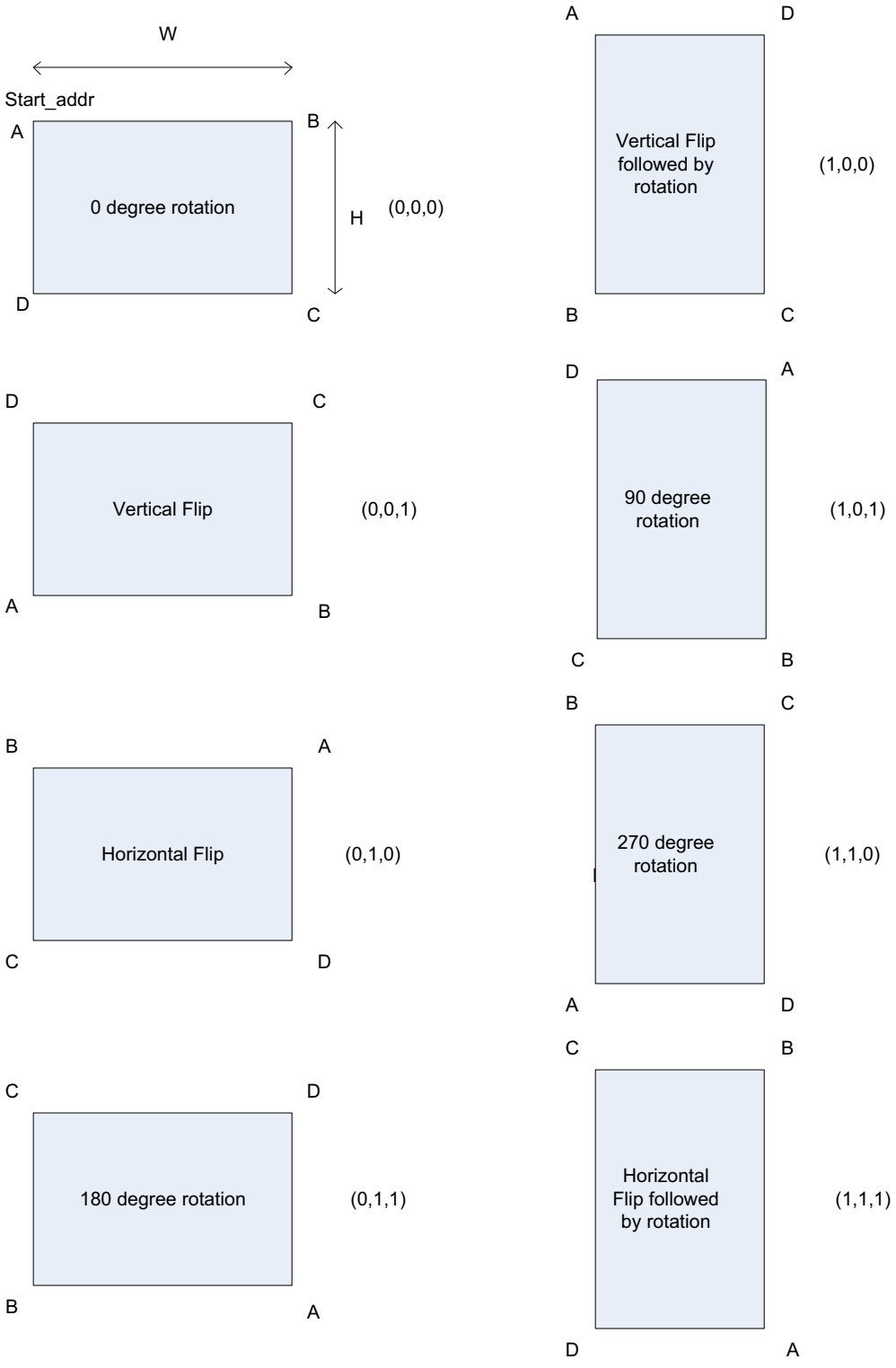
- B_WIN_OPTIONS
  - H_DIR
  - V_DIR
  - SCAN_COLUMN (“column” refers to surface pre-rotated columns)
- B_LINE_STRIDE
- B*_BUF_STRIDE
- B_START_ADDR_*
- B_ ADDR_H_OFFSET_*
- B_ ADDR_V_OFFSET_*

The following registers apply to post rotation:

- B_WIN_OPTIONS
  - H_FILTER*
  - V_FILTER*
- B_PRESCALED_SIZE
- B_SIZE
- B*_INITIAL_DDA
- B_DDA_INCREMENT

Because the memfetch unit is essentially walking the Y axis in memory, the majority of the window registers and associated logic remain unchanged. Therefore except for B_ADDR_H/V_OFFSET horizontal and vertical directions are with respect to post rotation.

Figure 87: Rotation Combinations



24.3.1.2 Line Buffer

The memfetch line buffer allows the display controller to retain one or more scan lines of window data locally so that no further reading of the scan line is required from memory. The scenarios where this is required are:

- Vertical scaling and filtering
  - Requires 3 scan lines of data for non-rotated memory fetches

    - Requires 5 scan lines of data for non-rotated memory fetches of packed data when compression is enabled
    - Requires 5 scan lines of data for rotated memory fetches of packed data
- Reading block linear data
    - Data is sent in 2 scan lines for non-rotated memory fetches
    - Data is sent in 4 scan lines for non-rotated memory fetches with compression enabled
    - Data is sent in 4 scan lines for rotated memory fetches

Data is received from the memory interface and steered to the proper location in the line buffer based on the thread ID in the tag FIFO. Data is read out of the line buffer and unpacked based on the pixel format. These pixels are placed in an output FIFO of memfetch to be sent to the window scaler logic in the display pipe.

In the vertical scaling and filtering scenario, up to five scan lines of data are required for the vertical scaler engine for it to generate the proper scaled output. In the block linear mode scenario, only a single scan line is output but the 32B returned data contains portions of two scan lines. To ensure adequate storage of RGB packed data format and steady bandwidth consumption to the display controller, line buffers are deep enough to hold up to five lines of maximum width of packed pixel data based on the maximum use-case supported by the given window.

Support for YUV planar rotation when reading data in BL 16Bx4 format requires a larger amount of data storage. Each plane of data will return 16B per color component. Due to scaling, it is required to keep a scan line of data in the buffer when fetching the next block of 16. This will require an additional byte of data. Therefore the buffer requirements for YUV planar are:

- Number of bytes per plane = 17 bytes
    - Minimum data fetch size = 16 bytes
    - Scaling requires additional scan-line remain
    - Pixel component depth per plane = 1 byte
- Number of planes = 3
- Line buffer width = Maximum width + 64 bytes
- Line buffer size = 17 bytes/plane* 3 planes/pixel * (maximum width+64) pixels per line
- Line buffer size for Y plane = (17 bytes/plane * 3 planes/pixel + 1 extra Y plane) * (maximum width+64) pixels per line

Since the line buffer storage requirement for YUV planar is close to three times (3X) the RGB packed format, the Tegra X1 display only supports buffering support for YUV planar rotation on 1 window. This restriction applies to YUV422 packed data as well due to the implementation of expanding the data to YUV444 in the line buffer. The other two windows will contain buffering to support non YUV planar rotation.

Below are the equations used to calculate buffer requirements for a given rotated video image.

- YUV444 (packed):
    - Scaling – 5 lines * (4Bpp*width + 64 bytes)
    - No scaling – 4 lines * (4Bpp*width + 64 bytes)
- YUV422 (packed):
    - Scaling – 9 lines*4Bpp (as data is expanded)* width + 9*64 bytes
    - No scaling – 8 *4Bpp(as data is expanded)*width + 8*64 bytes
- YUV Planer:
    - Scaling: (3 (number of surfaces)*17+1) * (width+ 64 bytes)
    - No scaling: 3 (number of surfaces)* 16 * (width + 64 bytes)

The line buffer helps to prevent refetching of data from memory. The line buffer must also account for invalid bytes fetched within a line due to address offset. This adds 64 bytes per line per plane (after aligning the line width in bytes to a multiple of 64).

The line buffer size requirement to prevent refetch depends on the use case. Based on the use case and method of packing/unpacking pixels stored in the line buffer, there will be an inherent pixel latency associated with the buffer. The memory controller can take advantage of this latency to temporarily stop sending pixel data to the display for cases such as changing the memory clock frequency.

Note that the line buffer can be written or read either with single or multiple lines per clock. For rotation use cases, line buffers load up to 16 lines at a time based on pixel format. Line buffers are read one or two lines at a time based on vertical scaling. The following tables and figures illustrate the line buffer size as well as inherent latency for various use cases for packed ARGB formats (4Bpp). For rotation use cases only block linear formats are supported. Remember that for rotation, block linear format is 16Bx4 lines.

Table 123: Non-Rotation Use Cases

Scenario	Vertical Scaling Enabled	Data Format	Line buffer size required to avoid refetch	Latency Tolerance Guaranteed by Line Buffer
A	NO	Pitched Linear	0	0 lines
B	NO	Block Linear	1 line	0 lines
C	YES	Pitched Linear	1 line	0 lines
D	YES	Block Linear	3 lines	0 lines
E	YES	Block Linear	5 lines (for compression enabled)	0 lines

Table 124: Rotation Use Cases

Scenario	Vertical Scaling Enabled	Data Format	Line buffer size required to avoid re-fetch	Latency Tolerance Guaranteed by Line Buffer
A	NO	Block Linear	4 line	1 line
B	YES	Block Linear	5 lines	0 lines
C	NO	YUV Planer	16 line	1 line
D	YES	YUV Planer	17 lines	0 lines

The latency tolerance refers to the number of lines available in the line buffer when the line buffer is full between when a given pixel is written to and read from the line buffer. For some of these use cases, there is a relatively long period of time after a pixel is written to the buffer before it is needed in the scan line. However, note that in every case where a scan line or more of data exists, the scan lines are being written at a rate of ½ or ¼ of the pixel clock rate since either 2 or 4 scan lines are being read at once from memory at a rate of 1 pixel per clock.

The fixed sizes of the Tegra X1 display buffers is given below. Each buffer size has a corresponding “maximum use-case” scenario from those listed above. For those scenarios that are not “maximum use-case”, the portion of the line buffer beyond what the scenario required can be used as latency buffering. Even the “maximum use-case” scenario can use a portion of the pixel latency as latency buffering.

Table 125: Window Class Line Buffer Sizes

Windows	Line Buffer Size (KB)	Maximum Use-Case	Maximum Surface Size
DisplayA WinAD	150	Rotation/Vertical Scaling/Block Linear/Planar	4096x2304
		Rotation/Vertical Scaling/Block Linear/Packed/Compression	4096x4096
DisplayA WinB/C	108	Rotation/Vertical Scaling/Block Linear/Packed/Compression	4096x4096
		Non-rotation/Vertical Scaling/Block Linear/Packed/Compression	4096x4096
DisplayB Win A/B/C	108	Non-rotation/Vertical Scaling/Block Linear/Compression	4096x4096
Window D/T	18	No scaling/non-rotation/non-block-linear/non-compression/RGBA only	4096x4096

To take advantage of the portion of the line buffer dedicated for handling latency MEMFETCH will send flags to the MC indicating when a programmable threshold is crossed. For each scenario, software must calculate the portion of the line buffer that can be used for latency buffering and set the threshold accordingly. Below is a list of the flags MEMFETCH will generate:

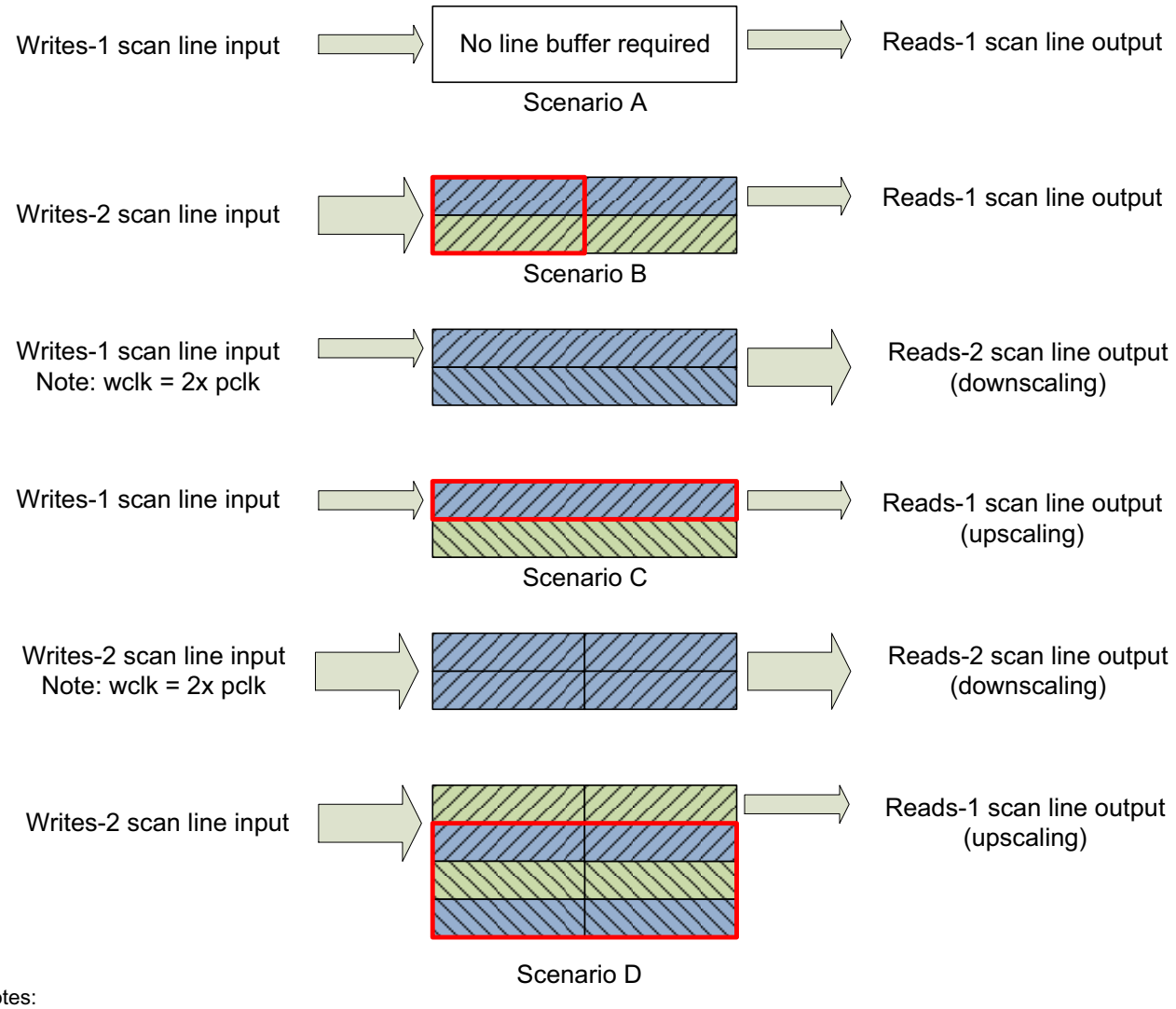
- SPOOLUP
  - Signal name: csr_display0a2mc_spool_up
  - Active from when data is requested during Vblank until active space
  - Active space starts with 1st pixel of window sent to display pipe
- DVFS readiness flag
  - Requires HWM threshold
  - Signal name: csr_display0a2mc_ready_for_latency_event
  - Active when:
  - Buffer exceeds HWM threshold
  - Window is disabled
- From the time all data is read for current frame until data is requested for subsequent frame
  - Deactivated with occurrence of underflow for the remainder of the frame
- LATENCY ALLOWANCE SCALE FACTOR buffering
  - Requires HWM/LWM thresholds
  - Signal names:
  - csr_display0a2mc_la_th_above_hi = level > HWM
    - deactivate with occurrence of underflow for remainder of frame
    - activate when window is disabled
  - csr_display0a2mc_la_th_below_lo = level < LWM
    - activate with occurrence of underflow for remainder of frame
    - deactivate when window is disabled

The programmable threshold will simply keep a running count of the number of bytes in the buffer and compare that value against the one in the threshold register.

Line buffering for planer YUV requires separate storage per plane. Each line is read by sending 1KB of Y requests followed by 1KB of U then 1KB of V requests. This is maintained until the entire line of YUV is read. Subsequent line requests are not issued for any component until all component requests have been issued for the current line. MEMFETCH does allow for changing the weights of the given requests per plane to help control the behavior of the requests per plane.

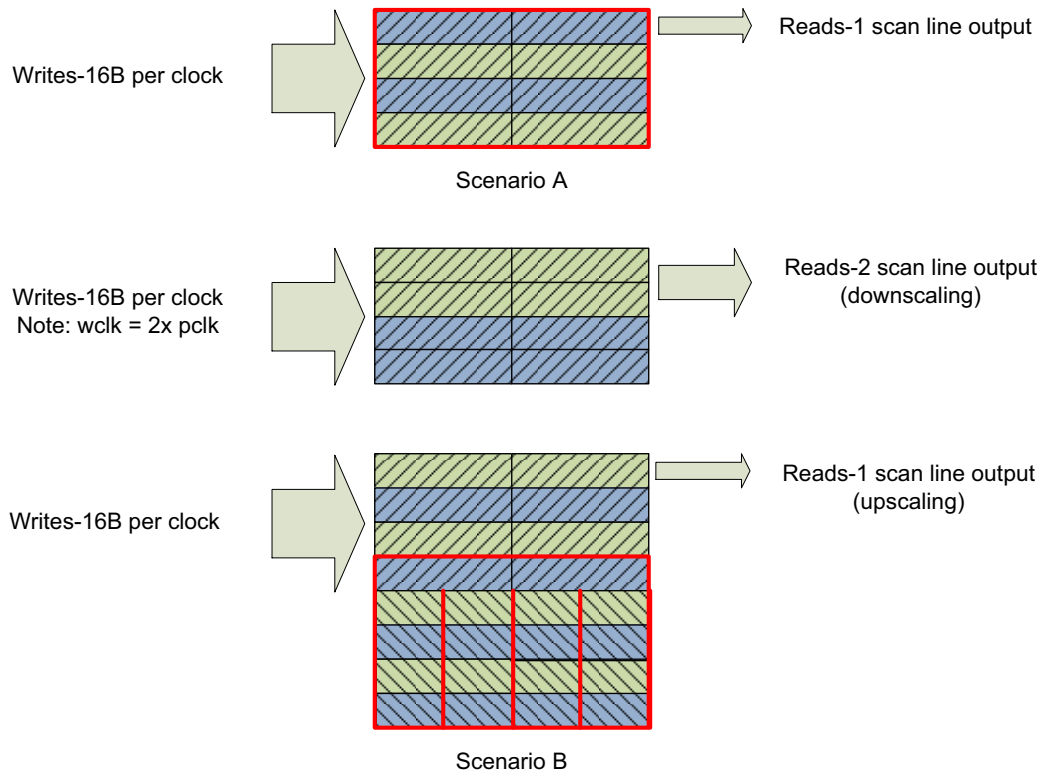
For non-rotated YUV planer use cases, the size of the line buffer is less than that of 4B ARGB since the pixel format is only 1.5B/pixel. However rotated planer YUV requires storage of 16B for each line per plane. Contact your NVIDIA AE should you need further support on this.

Figure 88: Non-Rotation Line Buffer Use Cases



Notes:
- Diagonal Lines represent the order in which data is written to the buffer (Two rows that have the same line pattern are written in parallel)
- Color represents the order in which data is read from the buffer (the same color indicates the rows are read in parallel)
- Red box indicates the line buffer size required to avoid refetch

Figure 89: Rotation Line Buffer Use Cases



Note:
- Diagonal lines Indicate the order in which data is written to the buffer.
  If the two rows have same line pattern, they are written in parallel.
- Color indicates read pattern
- Red boxes indicate line buffer size needed to avoid refetch
- Red lines indicate progress of filling line buffer as scan lines are read

24.3.1.3 Handling of Underflow

Display is an isochronous client that requires a steady bandwidth allocation. If data is not returned at a fairly constant rate and the line buffers run dry then data will not appear on the screen at the proper time. Eventually when data does start to flow to the display controller the image will appear shifted and there is no way for the image data to catch up to where it should be on the screen. This undesirable lack of guaranteed data is called underflow and it can happen in 2 forms:

- Short term underflow – pixels worth of missing data
- Long term underflow – scan-lines worth of missing data

For short term underflow it is possible for the display controller to catch up to the correct pixel by the start of the next scan-line by reading pixels during the horizontal blanking period. Short term underflow is being managed by Tegra display controller in this manner. For long term underflow it would be impossible for the display controller to throw out that much data. Short term underflow may arise when the memory controller is over-taxed based on a momentary surge in memory client requests. Long term underflow may happen when memory controller frequency changes are being made.

Tegra X1 display windows will address short term underflow similar to what is done in Tegra 3. Long term underflow is also handled in a manner similar to Tegra 3 during the active frame portion. Below is a description of Tegra X1 Underflow recovery improvements:

- Underflow recovery within a frame
  - Start counter of underflow pixels when underflow occurs
  - Last valid pixel is held at the beginning at the display pipeline for duration of the underflow

- Underflow pixels that eventually return to display are stored in line buffer and immediately read/discarded
- Operation continues (even during HBlank) until all underflow pixels have been discarded
- Once valid pixel is present at display pipe input memfetch operations return to normal mode
- Limit requests issued at the end of a frame if window is in underflow using the following method:

For cases when underflow continues to the end of the frame

- Add threshold counter (B_UFLOW_THRESHOLD)
- Track prescaled scanout
- Stop issuing frame requests when
  - prescaled scanout + threshold is > prescaled size

- Underflow recovery between frames
  - Add a frame bit to the tag FIFO to allow DP logic to determine which of the responses that are coming back are for the previous frame
  - Drop pixels from previous frame at 64B/pclk rate
  - Within display pipe, reset the underflowed pixel counters
  - Limit requests issued at the end of a frame if window is in underflow using method described above
  - Last pixel of previous frame is displayed

The Tegra X1 display handles cursor underflow in the following manner:

- Interface between memfetch and display pipeline senses when underflow occurs.
- For simple windows, the display pipe drives the last valid pixel for the duration of the underflow.
- When underflow occurs display pipeline drives a cursor alpha value of 0 for the remainder of the underflow period. This could be until the end of 1 or more scan lines.
- Memfetch will not drive valid to the display pipeline until the start of the scan line when data is available.
- Display pipeline will not assert ready to memfetch until the beginning of the scan line when data is available and memfetch is driving valid.
- Memfetch keeps track of number of clocks of missing data and when data does appear it reads out the data until the end of the scan line is reached.

It is important to highlight underflows for debug purposes. Current display units raise an interrupt to indicate that underflow has occurred. The Tegra X1 display continues to drive an interrupt when an underflow is detected. To further assist with highlighting underflows, memfetch drives a solid color from the window that experiences an underflow if the associated enable bit is active.

24.3.2 Display Stream Compression (DSC)

The DSC compression block takes pixel/control signals from the display and outputs compressed data to the DSI. The DSC block is compliant with VESA Display Stream Compression (DSC) Standard v1.1.

24.3.3 Color Decompression Engine (CDE)

The Color Decompression Engine (CDE) resides on the VIC and Display memory read interfaces. It transparently converts regular requests sent by the VIC or display unit to compressed requests, and decompresses data in the read responses on-the-fly before sending it back to the client. Compressed surfaces are created by the GPU. Refer to Chapter 25: Color Decompression Engine in this TRM for more information.

24.3.4 Interlaced Fields

To support set top boxes, the Tegra X1 display supports interlaced format, which is still prevalent in many television displays and broadcast networks. Interlaced format requires half the bandwidth of progressive and alternates sending fields containing odd and even lines of the original video frame.

The CEA Standard (CEA-861-E) defines these interlaced fields as FIELD1 and FIELD2. FIELD1 contains the even lines (line 0, 2, 4, etc.). FIELD2 contains the odd lines (line 1, 3, 5, etc.). From a consumer electronic display perspective, FIELD1 contains the top lines, and FIELD2 contains the bottom lines.

Memfetch contains start (base) addresses for both fields and when interlaced mode is enabled it will toggle between these base addresses every frame. Memfetch will continue to toggle between FIELD1 and FIELD2 fields until interlace mode is disabled. When interlace mode is disabled, memfetch only uses the normal (FIELD1) registers.

IMPORTANT: The first field that is fetched when interlace mode is based on HVTGEN field1/field2 field status. It is up to software to ensure that proper field being fetched is ready when activating interlace mode in memfetch.

No parameter other than start address and offset can change between fields such as size and stride.

To support double buffering of interlaced fields, software is required to update field1/field2 base addresses on a frame (not a field) boundary. Because the display is treating each field as a frame, software should make sure that updates only occur after an even number of frame SYNCPTS are received. It is strongly suggested that both field1 and field2 fields are ready in memory when performing the buffer swap to avoid coordinating field updates based on current field scanout.

24.3.5 Surface Formats

The following table lists the surface formats supported by the Tegra X1 device. The 32-bit RGBA formats that support compression are bolded.

Name (MSB..LSB)	Legacy Name (LSB..MSB)
T_P8	P8
T_A4R4G4B4	B4G4R4A4
T_A1R5G5B5	B5G5R5A
T_R5G6B5	B5G6R5
T_R5G5B5A1	AB5G5R5
T_A8R8G8B8	B8G8R8A8
T_A8B8G8R8	R8G8B8A8
T_U8_Y8__V8_Y8	CbYCrY422 YCbCr422
T_U8_Y8__V8_Y8_TRUE	UYVY422 YUV422
T_Y8___U8___V8_N420	YCbCr420P
T_Y8___U8___V8_N420_TRUE	YUV420P
T_Y8___U8___V8_N422	YCbCr422P
T_Y8___U8___V8_N422_TRUE	YUV422P
T_Y8___U8___V8_N422R	YCbCr422RP YCbCr422R
T_Y8___U8___V8_N422R_TRUE	YUV422RP YUV422R
T_A4B4G4R4	R4G4B4A4
T_A1B5G5R5	R5G5B5A
T_B5G5R5A1	AR5G5B5
T_X1R5G5B5	B5G5R5X1
T_R5G5B5X1	X1B5G5R5
T_X1B5G5R5	R5G5B5X1
T_B5G5R5X1	X1R5G5B5
T_B5G6R5	R5G6B5

Name (MSB..LSB)	Legacy Name (LSB..MSB)
T_X8R8G8B8	B8G8R8X8
T_X8B8G8R8	R8G8B8X8
T_Y8___U8___V8_N444	YCbCr444P
T_Y8___U8V8_N420	YCrCb420SP
T_Y8___V8U8_N420	YCbCr420SP
T_Y8___U8V8_N422	YCrCb422SP
T_Y8___V8U8_N422	YCbCr422SP
T_Y8___U8V8_N422R	YCrCb422RSP
T_Y8___V8U8_N422R	YCbCr422RSP
T_Y8___U8V8_N444	YCrCb444SP
T_Y8___V8U8_N444	YCbCr444SP
T_Y8___U8___V8_N444_TRUE	YUV444P
T_Y8___U8V8_N420_TRUE	YVU420SP
T_Y8___V8U8_N420_TRUE	YUV420SP
T_Y8___U8V8_N422_TRUE	YVU422SP
T_Y8___V8U8_N422_TRUE	YUV422SP
T_Y8___U8V8_N422R_TRUE	YVU422RSP
T_Y8___V8U8_N422R_TRUE	YUV422RSP
T_Y8___U8V8_N444_TRUE	YVU444SP
T_Y8___V8U8_N444_TRUE	YUV444SP

24.3.6 Horizontal Filter

Memfetch and scaler provide 6 adjacent pixels per clock, along with DDA fraction indicating filter kernel phase. Fraction selects among 16 sets of coefficients for filter kernel.

Note: *When scaling RGBA values, the alpha channel is never filtered, so the output is a point-sampled result.*

24.3.7 Vertical Filter

When enabled, memfetch provides 2 adjacent lines in parallel, which are independently horizontally filtered before being sent to vertical filter. The fractional DDA from vertical scaler (indicating filter phase) selects among 16 filter kernel coefficients. For coefficient entry N, the weights are N and 1-N.

Note: *When scaling RGBA values, the alpha channel is never filtered, so the output is a point-sampled result.*

24.3.8 Color Space Converter

The full-featured memfetch windows A/B/C support color space conversion. The color space converter converts YCbCr and YUV to RGB. It can also do limited RGB to RGB conversions. This feature is different from the CMU CSC, which is post-blending.

24.3.9 Color Palette/LUT

To support gamma adjustment and palletized formats, each window A/B/C has a color palette look-up table (CP LUT) with three 256x8-bit tables, one each for red, green, and blue. The three tables are independent and can represent arbitrary functions. For 4-bit palette modes, only the bottom 16 entries are used.

Note that the LUTs, and subsequent display pipeline, do not have the resolution needed for linear RGB. Thus the CP LUTs cannot really be used to “un-gamma” input data. They can be used for minor adjustments to the tone curve.

24.3.10 Digital Vibrance

Using the same algorithm as NVIDIA GeForce GPUs, digital vibrance can increase the saturation of colors.

24.3.11 Cursor

The cursor supports two pixel formats:

- Legacy 2-bit-per-pixel, where the four values are background color, foreground color, transparent, and invert. Foreground and background color are programmable via registers with 24-bit RGB values. Transparent means cursor is invisible and desktop shows through. Invert means desktop RGB is inverted (red’ = 255 – red, etc.). Pixels are packed 4 per byte, and each line has 16 bytes pitch, regardless of whether it is 32 or 64 pixels wide.
  - Cursor sizes of 32x32 or 64x64
- 32-bpp RGBA pitch linear
  - Full color with alpha
  - Cursor sizes of:
    - 32x32
    - 64x64
    - 128x128
    - 256x256

The blending required for the alpha cursor will occur near the end of the blending pipeline (on top of windows A/B/C/D).

Note: *Legacy cursor does not support per-pixel or global alpha.*

The cursor can be clipped to either the display, or window A, B, or C. Thus the cursor active region for blending must be computed according to that mode. In regions where cursor is clipped, or outside the cursor entirely, blending behaves as if cursor is disabled (i.e., previous blend stage passes through.)

Note: *The Tegra X1 display cursor supports partial clipping but does not support trivial reject of cursor when the cursor is completely outside of the window region. It is the responsibility of software to ensure that the cursor is at least partially visible.*

24.3.11.1 Legacy Support for Blending

The Tegra X1 display controller supports cursor modes used in previous generations of the Tegra display. Legacy cursor blending uses 2 bits per pixel and programming for this mode will remain unchanged.

24.3.11.2 32-bpp RGBA Cursor Mode

In 32-bpp RGBA cursor mode, legacy adaption is disabled. Cursor RGBA is blended with the last stage of the window blending unit. Cursor alpha value controls this blending.

24.3.11.3 Interlaced Cursor Support

To support interlaced output, the cursor performs a progressive-to-interlaced format conversion of the cursor image in memory. Since the cursor is static, no low pass filtering is necessary. The display cursor reads every other line from the square cursor thus reading only half of the lines. Based on the cursor destination Y position and the field, the Tegra X1 display starts reading line 0 or 1 from memory. The decision on whether to start on cursor line 0 or 1 for the first or second field is left to software,

giving more flexibility to manipulate the cursor image and set the offset for a given field. An interlace sequence example for cursor is shown below:

1. Program CURSOR_INTERLACE_ENABLE=1
2. Program CURSOR_INTERLACE_START =1
3. The cursor reads every other line from memory starting at line 1 and stores it to sequential panel lines
4. On the next “frame”, the cursor toggles the line start to 0

   CURSOR_INTERLACE_STATUS = 0
5. The cursor reads every other line from memory starting at line 0 and stores it to sequential panel lines

Repeat steps 2 through 5 in the above sequence when the cursor position is updated and set the cursor start line appropriately.

The cursor interlace control register is shown below. When the CURSOR _INTERLACE_START bit is written, hardware should update the corresponding CURSOR_INTERLACE_STATUS bit to this value. This allows the start line to track updates to the cursor position. The CURSOR_INTERLACE_FIELD1/2_VOFF_INCR bits determine if V_CURSOR_POSITION is incremented by one for that field. The field status is tracked by the hardware cursor logic using HVTGEN interlace field status information.

Table 126: Cursor Interlace Control Register

Field Name	Setting
CURSOR_INTERLACE_ENABLE	Interlace Enable 0=disable 1=enable
CURSOR_INTERLACE_FIELD1_VOFF_INCR	Field1 v position Incr Enable 0=disable 1=enable
CURSOR_INTERLACE_FIELD2_VOFF_INCR	Field2 v position Incr Enable 0=disable 1=enable
CURSOR _INTERLACE_START	Starting line. 0=line 0, 1=line 1
CURSOR_INTERLACE_STATUS	Current line start. Read only. 0=line 0, 1=line 1

There are two possibilities for cursor vertical position in a progressive frame – odd or even line. When breaking a progressive frame into interlaced fields, it is important to know what cursor line number to start on per field when the cursor vertical position starts on an odd progressive line and when it is on an even progressive line. Consider the following examples:

Example 1 cursor starts on line 10:

- V_CURSOR_POSITION = 5
- First line of cursor is at progressive line offset 10

  Corresponds to FIELD1 line 5
- Second line of cursor is at progressive line offset 11

  Corresponds to FIELD2 line 5

Example 2 cursor starts on line 1:

- V_CURSOR_POSITION = 0
- First line of cursor is at progressive line offset 1

  Corresponds to FIELD2 line 0
- Second line of cursor is at progressive line offset 2

  Corresponds to FIELD1 line 1

  Requires setting CURSOR_INTERLACE_FIELD1_VOFF_INCR=1

The following table describes programming mechanism for line start based on progressive frame vertical cursor offset for interlaced fields FIELD1 and FIELD2.

Interlace field	Progressive frame cursor Y offset	
	Odd	**Even**
First field (FIELD1)	CURSOR_INTERLACE_START = Line 1 V_CURSOR_POSITION = INT(Progressive_cursor_y/2) + 1	CURSOR_INTERLACE_START = Line 0 V_CURSOR_POSITION = INT(Progressive_cursor_y/2)
Second field (FIELD2)	CURSOR_INTERLACE_START = Line 0 V_CURSOR_POSITION = INT(Progressive_cursor_y/2)	CURSOR_INTERLACE_START = Line 1 V_CURSOR_POSITION = INT(Progressive_cursor_y/2)

Note: *Programming of CURSOR_INTERLACE registers listed above must line up with the interlaced field in the table. Therefore the register should be written (armed) on the previous field for it to activate on the appropriate field.*

24.3.12 Blending

The Tegra X1 display uses weighted blending between windows. Legacy cursor blending must be disabled when full-color alpha-blended cursor is in use.

In Tegra X1 devices, the display blender has six sequential blending stages. Each window is assigned a depth value. The windows are sorted by depth and assigned to a blend stage. The cursor is the last (top) stage. Each stage combines its pixels with the previous stage according to programmed equations.

Figure 90: Tegra X1 Blender Window Stages

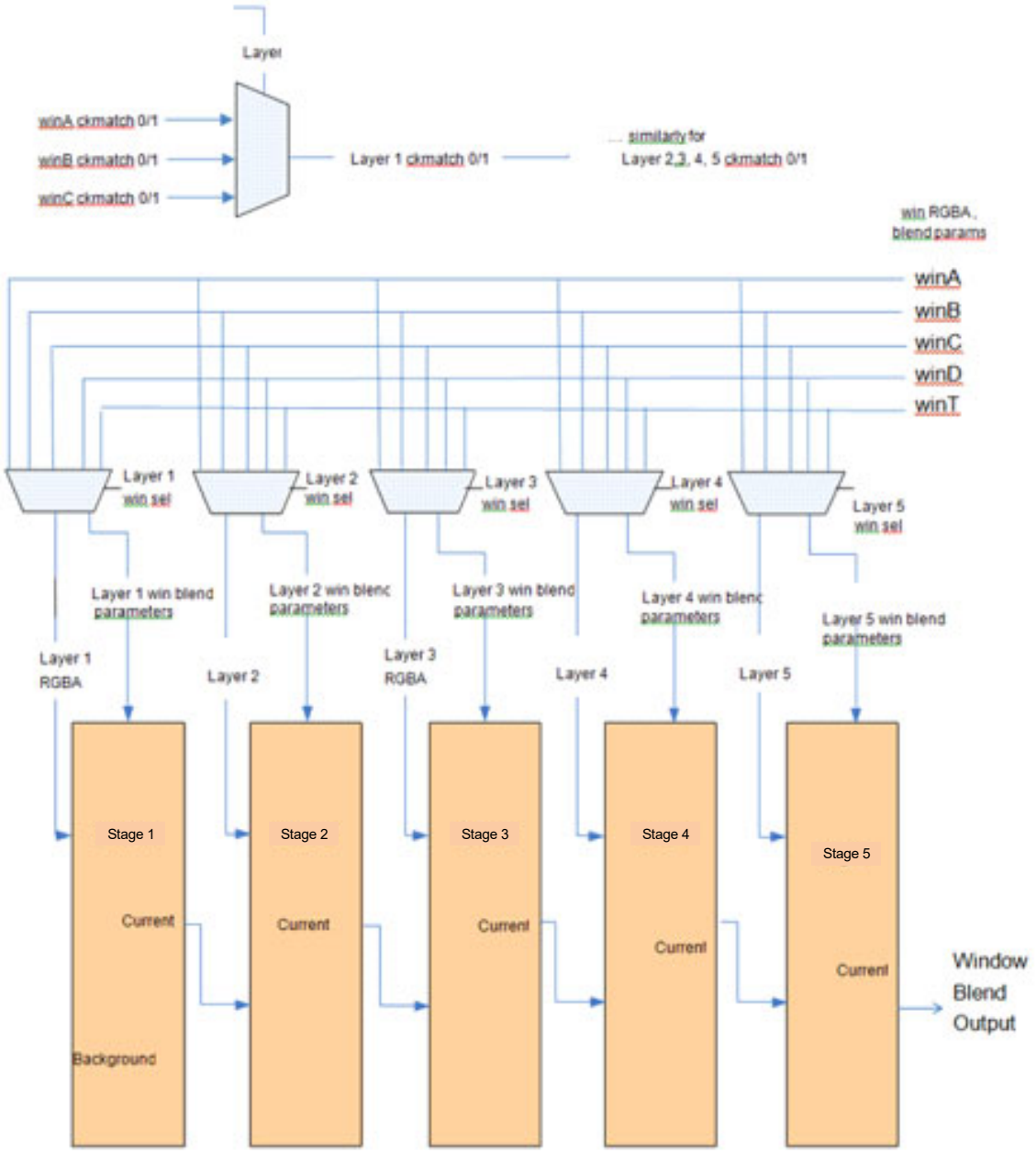
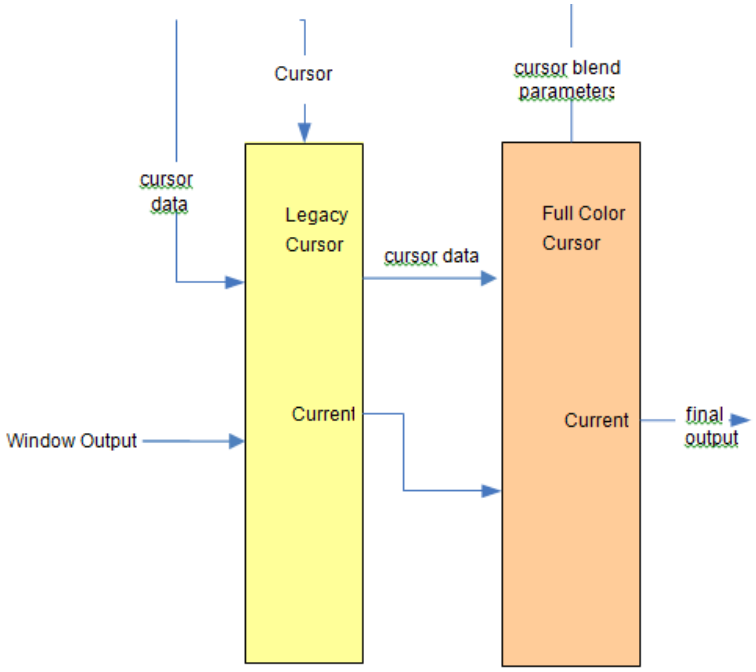


Figure 91: Tegra X1 Blender Cursor Stages



24.3.12.1 Blend Algorithm

Each blending stage requires:

- The current color from the previous stage (often called “Destination”)
- The surface color from the assigned window (often called “Source”)
- Window enable (that is, whether windows cover the current pixel)
- Blend parameters from the per-window registers

Each window has the following registers (where x is A, B, C, D, or T). The cursor settings are also described below.

```
windowx.Z  - an 8 bit z value
windowx.K1 - an 8 bit constant alpha value
windowx.K2 - an 8 bit constant alpha value
windowx.blend_bypass - a boolean

windowx.srcFactC_Match_Select - an enum with the following options:
            K1                      windowx.K1
            K1_TIMES_DST            windowx.K1*currentA
            NEG_K1_TIMES_DST        1 - (windowx.K1*currentA)
            K1_TIMES_SRC            windowx.K1*layerxA
            ZERO                    0

windowx.dstFactC_Match_Select - an enum with the following options:
            K1                      windowx.K1
            K2                      windowx.K2
            K1_TIMES_DST            windowx.K1*currentA
            NEG_K1_TIMES_DST        1 - (windowx.K1*currentA)
            NEG_K1_TIMES_SRC        1 - (windowx.K1*layerxA)
            ZERO                    0
```

```
            ONE                     1.0 (0xff)

windowx.srcFactA_Match_Select - an enum with the following options:
            K1                      windowx.K2
            K2                      windowx.K2
            ZERO                    0

windowx.dstFactA_Match_Select - an enum with the following options:
            K2                      windowx.K2
            ZERO                    0
            NEG_K1_TIMES_SRC        1 - (windowx.K1*layerxA)
            ONE                     1.0 (0xff)
```

Tegra X1 devices support color key; however, color key is not described in this document. The “Match” blend registers above apply to cases where color key is disabled as well as to cases where color key is enabled but the key color does not match. There is an additional set of “NoMatch” registers for the color key enabled but not matching cases.

The windows are sorted according to Z (Depth) value and assigned to blending stages 1 to 5. Cursor is layer 6. Each layerx (where x is 1 ..6) operates as follows:

Inputs

```
layerxRGBA - the RGBA color from the corresponding window pipeline
windowx.* - window registers from window associated with this layer
currentxRGBA - output from the preceding layer_blendx unit
```

Layer0 only needs the BackgroundColorRGBA (also known as Border Color) as input

Layer 6 (cursor) gets layerxRGBA from the cursor surface and gets windowx.* from special cursor.* registers as follows:

```
windowx.key_Select = NONE
windowx.K1 = cursor.K1
windowx.K2 = cursor.K2
windowx.dstFactC_Match_Select = cursor.dstFactC_Select
windowx.srcFactC_Match_Select = cursor.srcFactC_Select
```

The following are “don’t cares” but can be set as shown:

```
windowx.dstFactA_Match_Select   = K2
windowx.srcFactA_Match_Select   = K2
```

Outputs

```
outputRGBA – 32-bit blended RGBA color
```

Local State

```
match - boolean
dstFactC_Select - enum
srcFactC_Select - enum
dstFactA_Select - enum
srcFactA_Select - enum
dstFactC - 8 bit alpha value
srcFactC - 8 bit alpha value
dstFactA - 8 bit alpha value
srcFactA - 8 bit alpha value
```

Operation

At each pixel of the output raster, the blending is performed. Only a subset of windows touch (cover) each pixel, because windows have an arbitrary size and position on the “desktop” raster.

layer_blend0 is trivial:

```
outputRGBA = BackgroundColorRGBA
```

layer_blendx (where x is 1 through 5) does the following:

```
    - calculate fact??_Select:
if [match] then
        srcFactC_Select = windowx.srcFactC_Match_Select
        dstFactC_Select = windowx.dstFactC_Match_Select
        srcFactA_Select = windowx.srcFactA_Match_Select
        dstFactA_Select = windowx.dstFactA_Match_Select
else
        srcFactC_Select = windowx.srcFactC_NoMatch_Select
        dstFactC_Select = windowx.dstFactC_NoMatch_Select
        srcFactA_Select = windowx.srcFactA_NoMatch_Select
        dstFactA_Select = windowx.dstFactA_NoMatch_Select

    - calculate srcFactC:
        switch (srcFactC_Select) {
            case K1:                    srcFactC = windowx.K1
            case K1_TIMES_SRC:          srcFactC = windowx.K1*layerxA
            case K1_TIMES_DST:          srcFactC = windowx.K1*currentA
            case NEG_K1_TIMES_DST:      srcFactC = 1 - (windowx.K1*currentA)
            case ZERO:                  srcFactC = 0
            case ONE:                   srcFactC = 1
        }

    - calculate dstFactC:
        switch (dstFactC_Select) {
            case K1:                    dstFactC = windowx.K1
            case K2:                    dstFactC = windowx.K2
            case ZERO:                  dstFactC = 0
            case ONE:                   dstFactC = 1
            case K1_TIMES_DST:          dstFactC = windowx.K1*currentA
            case NEG_K1_TIMES_SRC:      dstFactC = 1 - (windowx.K1*layerxA)
            case NEG_K1_TIMES_DST:      dstFactC = 1 - (windowx.K1*currentA)
            case NEG_K1:                dstFactC = 1 - windowx.K1
        }

    - calculate srcFactA:
        switch (srcFactA_Select) {
            case K1:                    srcFactA = windowx.K1
            case K2:                    srcFactA = windowx.K2
            case ZERO:                  srcFactA = 0
        }
```

```
- calculate dstFactA:
    switch (dstFactA_Select) {
        case K2:                    dstFactA = windowx.K2
        case ZERO:                  dstFactA = 0
        case ONE:                   dstFactA = 1
        case NEG_K1_TIMES_SRC:      dstFactA = 1 - (windowx.K1*layerxA)
    }
```

If the window or cursor for this layer does NOT touch this pixel, then:

```
        outputRGBA = currentRGBA
    else if windowx.blend_bypass is TRUE then
        outputRGBA = layerxRGBA
    else
        outputR = (srcFactC * layerxR) + (dstFactC * currentR)
        outputG = (srcFactC * layerxG) + (dstFactC * currentG)
        outputB = (srcFactC * layerxB) + (dstFactC * currentB)
        outputA = (srcFactA * layerxA) + (dstFactA * currentA)
```

Note: windowx.blend_bypass allows bypassing the entire blend operation and is intended to save power.

For the cursor layer, there is no need to calculate the following:

```
    outputA - not needed
    dstFactA - not needed
    srcFactA - not needed
```

Final Result

The outputRGB from layer_blend6 is the pixel value sent to the display.

24.3.12.2 Example Use Cases

The examples below assume 3 windows, not 5, for simplicity.

```
                Eye is looking down from here
                           i
                           ^
layer 4                 -----              cursor
layer 3              -------               windowC
layer 2                   ----------       windowB
layer 1           ----------------------    windowA
layer 0           ----------------------    background
```

Common Settings

If not mentioned in a use case, assume registers have the following values:

```
BackgroundColorRGBA = background color

key0.MinRGBA = (don't care)
key0.MaxRGBA = (don't care)

key1.MinRGBA = (don't care)
key1.MaxRGBA = (don't care)

windowA.Z  = 3
```

```
windowB.Z  = 2
windowC.Z  = 1

window*.K1 = 0
window*.K2 = 0
window*.key_Select = NONE
window*.dstFactC_Match_Select = K1
window*.srcFactC_Match_Select = K1
window*.dstFactC_NoMatch_Select = K1
window*.srcFactC_NoMatch_Select = K1
window*.dstFactA_Match_Select = K2
window*.srcFactA_Match_Select = K2
window*.dstFactA_NoMatch_Select = K2
window*.srcFactA_NoMatch_Select = K2
```

If the cursor uses non-premultiplied alpha with fade factor F:

```
cursor.K1 = F
cursor.dstFactC_Select = NEG_K1_TIMES_SRC
cursor.srcFactC_Select = K1_TIMES_SRC

result = ((1.0-(F*cursorA)) * layer3RGB) + (F*cursorA * cursorRGB)
```

If the cursor uses premultiplied alpha with fade factor F:

```
cursor.K1 = F
cursor.dstFactC_Select = NEG_K1_TIMES_SRC
cursor.srcFactC_Select = K1

result = ((1.0-(F*cursorA)) * layer3RGB) + (F         * cursorRGB)
```

Below "use for fade" indicates that the value of windowx.K1 can be used to fade the window from invisible (transparent) at windowx.K1=0x00 to fully visible at windowx.K1=0xff.

Alpha Blend

```
layer 4                -----              cursor
layer 3            ccccccc                windowC non-premul
layer 2                 bbbbbbbbbb        windowB premul
layer 1         aaaaaaaaaaaaaaaaaaaaa     windowA no alpha
layer 0        gggggggggggggggggggggggg   background

result         gaaaccccccccbbbbbbbbbaaaag
                     ^    ^ ^
                     |    | |
                     |    | +---- b blended with a
                     |    +------ c blended with (b blended with a)
                     +---------- c blended with a
```

Window A has RGB (alpha not used)

Window B has premultiplied alpha RGBA for blending with A

Window C has non-premultiplied alpha RGBA for blending with A and B

Blend Equations

```
layer0RGBA = BackgroundColorRGBA
layer1RGB  = windowA color
layer1A    = don't care
layer2RGB  =             windowB.RGB  +  (1.0 - windowB.A) * layer1RGB
layer2A    = don't care
layer3RGB  = windowC.A * windowC.RGB  +  (1.0 - windowC.A) * layer2RGB
layer3A    = don't care
layer4RGB  = blend with cursor
```

Register Settings

```
windowA.K1 = 0xff  (use for fade)
windowA.key_Select = NONE
windowA.srcFactC_Match_Select = K1
windowA.dstFactC_Match_Select = ZERO

windowB.K1 = 0xff  (use for fade)
windowB.key_Select = NONE
windowB.srcFactC_Match_Select = K1  (K1_TIMES_SRC for non-premul)
windowB.dstFactC_Match_Select = NEG_K1_TIMES_SRC

windowC.K1 = 0xff  (use for fade)
windowC.key_Select = NONE
windowC.srcFactC_Match_Select = K1_TIMES_SRC    (K1 for premul)
windowC.dstFactC_Match_Select = NEG_K1_TIMES_SRC
```

Note: *Window B and window C can each be either premul or non-premul.*
Use windowx.K1 to fade the window (0=transparent, 0xff=opaque).

Opaque Layer with Fade

```
layer 4                -----              cursor
layer 3            ccccccc                windowC non-premul
layer 2                 bbbbbbbbbb        windowB no alpha
layer 1         aaaaaaaaaaaaaaaaaaaaa     windowA no alpha
layer 0        gggggggggggggggggggggggg   background

result         gaaaccccccbbbbbbbbaaaag
                     ^   ^ ^
                     |   | |
                     |   | +---- b blended with a using L
                     |   +------ c blended with (b blended with a using L)
                     +---------- c blended with a
```

Same as case 1, but window B (or A or C or any combination) contains no alpha values (or garbage alpha values). There is a constant L which is used to fade the window.

Blend Equations

Note: *"..." means "same as the last use case"*

```
...
layer2RGB  = L * windowB.RGB  +  (1.0 - L) * layer1RGB
layer2A    = don't care
...
```

Register Settings

```
...
windowB.K1 = L
windowB.K2 = 1.0 - L
windowB.srcFactC_Match_Select = K1
windowB.dstFactC_Match_Select = K2
...
```

Window B's Alpha is in Window A

```
layer 4                -----              cursor
layer 3            ccccccc                windowC premul
layer 2                 bbbbbbbbbb        windowB no alpha
layer 1         aaaaaaaaaaaaaaaaaaaaa     windowA alpha for blend w/ b
layer 0        gggggggggggggggggggggggg   background

result         gaaaccccccccbbbbbbbbbaaaag
                     ^   ^ ^
                     |   | |
                     |   | +---- b blended with a using alpha from a
                     |   +------ c blended with (")
                     +---------- c blended with a
```

Same as case 1, but window B contains no alpha values (or garbage alpha values), and the alpha is needed for use from window A to blend window B with window A.

Blend Equations

```
layer0RGBA = BackgroundColorRGBA
layer1RGB  = windowA color
layer1A    = windowA alpha
layer2RGB  = windowA.A * windowB.RGB + (1.0 - windowA.A) * layer1RGB
layer2A    = don't care
layer3RGB  = windowC.RGB + (1.0 - windowC.A) * layer2RGB
layer3A    = don't care
layer4RGB  = blend with cursor
```

Register Settings

```
windowA.K1 = 0xff  (use for fade)
windowA.K2 = 0xff
windowA.key_Select = NONE
windowA.srcFactC_Match_Select = K1
windowA.dstFactC_Match_Select = ZERO
windowA.srcFactA_Match_Select = K2
windowA.dstFactA_Match_Select = ZERO

windowB.K1 = 0xff  (use for fade)
windowB.key_Select = NONE
```

```
windowB.srcFactC_Match_Select = NEG_K1_TIMES_DST
windowB.dstFactC_Match_Select = K1_TIMES_DST

windowC.K1 = 0xff  (use for fade)
windowC.key_Select = NONE
windowC.srcFactC_Match_Select = K1  (K1_TIMES_SRC for non-premul)
windowC.dstFactC_Match_Select = NEG_K1_TIMES_SRC
```

There is no way to do this for B and C at the same time (unless they are known to not overlap).

If windowA.A is the opacity of window C, then use the following blend equation and register settings:

Blend Equation (windowA.A is the opacity of window C)

```
layer2RGB  = (1.0 - windowA.A) * windowB.RGB  +  windowA.A * layer1RGB
```

Register Settings (windowA.A is the opacity of window C)

```
windowB.srcFactC_Match_Select = NEG_K1_TIMES_DST
windowB.dstFactC_Match_Select = K1_TIMES_DST
```

If windowA.A is the transparency of window C, then use the following blend equation and register settings:

Blend Equation (windowA.A is the transparency of window C)

```
layer2RGB  = windowA.A * windowB.RGB  +  (1.0 - windowA.A) * layer1RGB
```

Register Settings (windowA.A is the transparency of window C)

```
windowB.srcFactC_Match_Select = K1_TIMES_DST
windowB.dstFactC_Match_Select = NEG_K1_TIMES_DST
```

The above assumes non-premul alpha. Premul alpha would not make sense in this case.

Window C's Alpha is in Window A

```
layer 4                -----              cursor
layer 3           ccccccc                 windowC no alpha
layer 2                bbbbbbbbbb         windowB premul alpha
layer 1         aaaaaaaaaaaaaaaaaaaaa     windowA alpha for blend w/ c
layer 0        gggggggggggggggggggggggg   background

result         gaaaccccccccbbbbbbbbbaaaag
                     ^   ^ ^
                     |   | |
                     |   | +---- b blended with a
                     |   +------ c blended with (") using alpha from a
                     +---------- c blended with a using alpha from a
```

Same as case 1, but window C contains no alpha values (or garbage alpha values), and the alpha is needed for use from window A to blend window C with window A and window B.

Blend Equations

```
layer0RGBA = BackgroundColorRGBA
layer1RGB  = windowA color
layer1A    = windowA alpha
layer2RGB  = windowB.RGB              +  (1.0 - windowB.A) * layer1RGB
layer2A    = don't care
layer3RGB  = windowA.A * windowC.RGB  +  (1.0 - windowA.A) * layer2RGB
```

```
layer3A    = don't care
layer4RGB  = blend with cursor
```

Register Settings

```
windowA.K1 = 0xff  (use for fade)
windowA.K2 = 0xff
windowA.key_Select = NONE
windowA.srcFactC_Match_Select = K1
windowA.dstFactC_Match_Select = ZERO
windowA.srcFactA_Match_Select = K2
windowA.dstFactA_Match_Select = ZERO

windowB.K1 = 0xff  (use for fade)
windowA.K2 = 0xff
windowB.key_Select = NONE
windowB.srcFactC_Match_Select = K1   (K1_TIMES_SRC for non-premul)
windowB.dstFactC_Match_Select = NEG_K1_TIMES_SRC
windowB.srcFactA_Match_Select = ZERO
windowB.dstFactA_Match_Select = K2

windowC.K1 = 0xff  (use for fade)
windowC.key_Select = NONE
windowC.srcFactC_Match_Select = NEG_K1_TIMES_DST
windowC.dstFactC_Match_Select = K1_TIMES_DST
```

24.3.13 PRISM2

This section describes the enhancements to PRISM2. The PRISM function was formerly known as Smart Dimmer, and so the abbreviation SD is commonly used in register names for PRISM2.

Key features include:

- Programmable statistics window
- Separate crush/adaptation
- Programmable soft clipping
- Increased pixel count limits
- Run per-frame on VPULSE2
- Enhanced phase-in (smooth K0)

24.3.13.1 CMU

The CMU impacts PRISM2 by allowing SD to run in sRGB-style gamma space, which is more suitable than the arbitrary tone curve of the display device.

24.3.13.2 Programmable Statistics Window

This feature allows statistics to be gathered over a subset of the active display. It is useful when part of the display is not interesting for PRISM2 statistics purposes, such as letterbox or border around a video. Without this feature, the border would influence the histogram and therefore the brightness, giving suboptimal results for the video.

The programming interface is a control bit SD_WINDOW_ENABLE in SD_CONTROL register to enable the feature, and two registers containing position and size of the window. When SD_CONTROL == DISABLE (the default), the position/size registers are ignored, and the entire active display region is used, like in Tegra 3.

The coordinate system is relative to display active area – (0,0) is upper left corner of display active. The active region will have upper left corner (H_POSITION, V_POSITION), and lower right corner (H_POSITION + SD_WIN_H_SIZE – 1, V_POSITION + V_SIZE – 1) inclusive. Note this display coordinate system does not take into account surface rotation or device rotation. It is the same coordinate system use by window position (e.g., B_H_POSITION, B_V_POSITION).

The limitation on minimum pixel count of 2^16 must be obeyed.

The programmable window has no direct effect on enhancement – every pixel of display active area is enhanced.

24.3.13.3 Separate Crush/Adaptation

As in earlier Tegra devices, the AGGRESSIVENESS register controlled both the maximum crushed pixel percentage, and the lowest K (maximum gain). This feature separates them and allows software to separately control lowest K (maximum gain). AGRESSIVENESS continues to control the maximum crushed pixel percentage.

The programming interface is a control bit K_LIMIT_ENABLE, and a register SD_K_LIMIT.

As in earlier Tegra devices, the initial K (as determined by accumulating histogram bins until maximum crush pixel percentage is reached). When K_LIMIT_ENABLE=ENABLE, the initial K is clamped to SD_K_LIMIT rather than default “lowest_K”. When K_LIMIT_ENABLE = DISABLE (the default), SD_K_LIMIT register is ignored and the default “lowest_L” are used based on AGGRESSIVENESS.

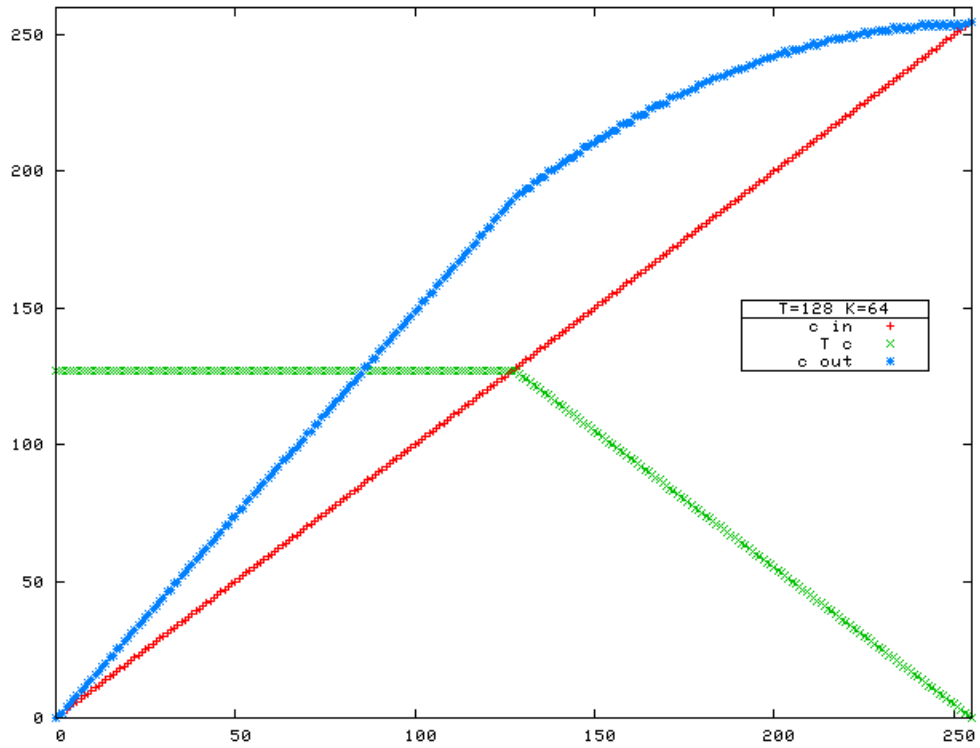
24.3.13.4 Programmable Soft Clipping

Soft clipping is employed to avoid harsh clipping of amplified high-value pixels, and resulting loss of detail in bright areas. The gain applied to high-valued pixels is reduced gradually from 1/K to 1.0, for pixels above a threshold of 128. The threshold is fixed at 128, and soft clipping is always enabled. This feature gives more software control.

Since soft clipping is performed after statistics, and it affects the brightness of the image, the statistics and therefore the backlight brightness will be incorrect. This can somewhat compensated for in the backlight transfer-function and/or enhancement value lookup tables.

The Tegra X1 transfer function is shown here, for K=1.5. The T_c plot is the weight applied to fractional portion of K, and goes from 127 to 0, representing 1 to 0.

Figure 92: Default Threshold=128 at K=1.5



With programmable thresholds, other curves are possible. For example, Figure 93 and Figure 94 show more gradual curves using threshold=64 and 0 respectively.

Figure 93: Threshold=64 at K=1.5

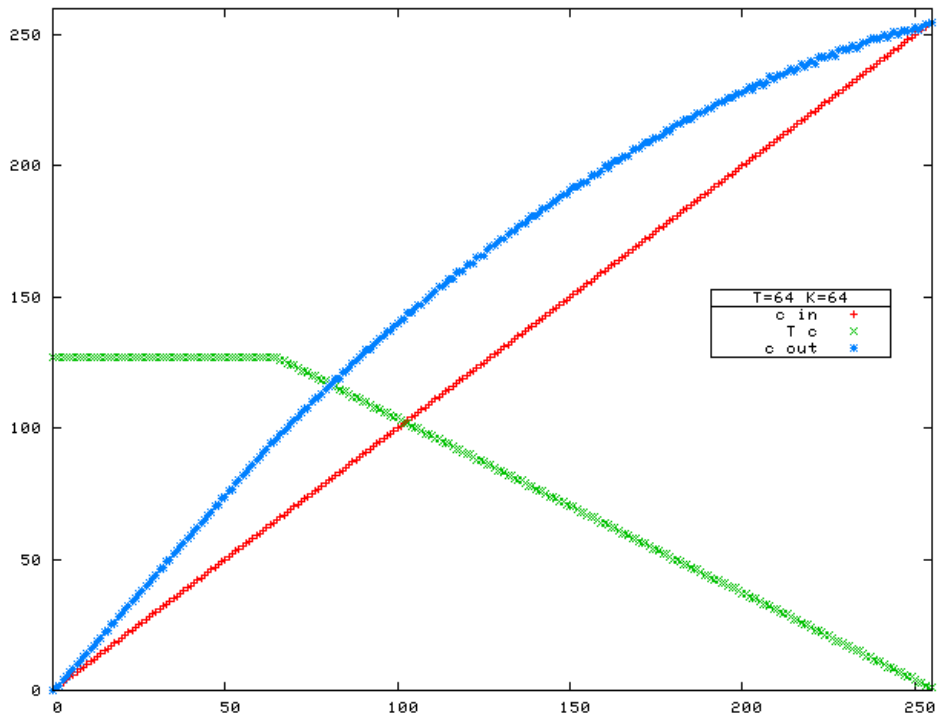
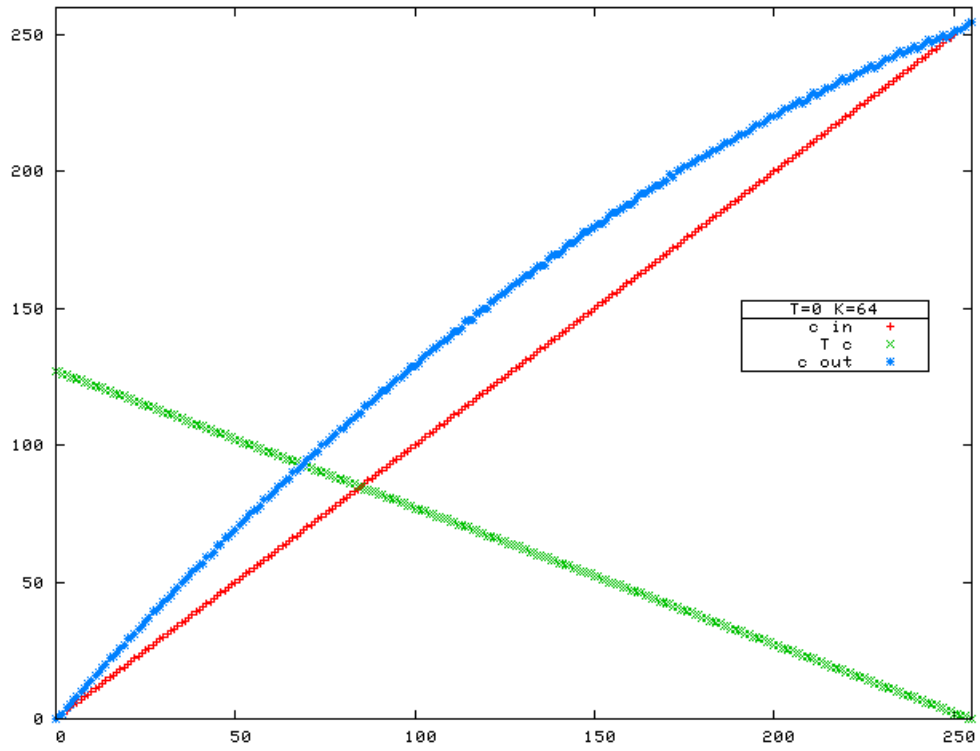


Figure 94: Threshold 0 at K=1.5



Thresholds too high are not usable, because the soft clipping does not kick in early enough to prevent clipping as K grows; note in Figure 95 (threshold 170 at K=1.5) the value of c_out rises about 255. The curve is also not monotonically increasing. Software would need to reduce maximum K to 1.25 or so for such a threshold. The following figure shows threshold of 170 with K = 1.25.

Figure 95: Overflowing at Threshold=170 at K=1.5

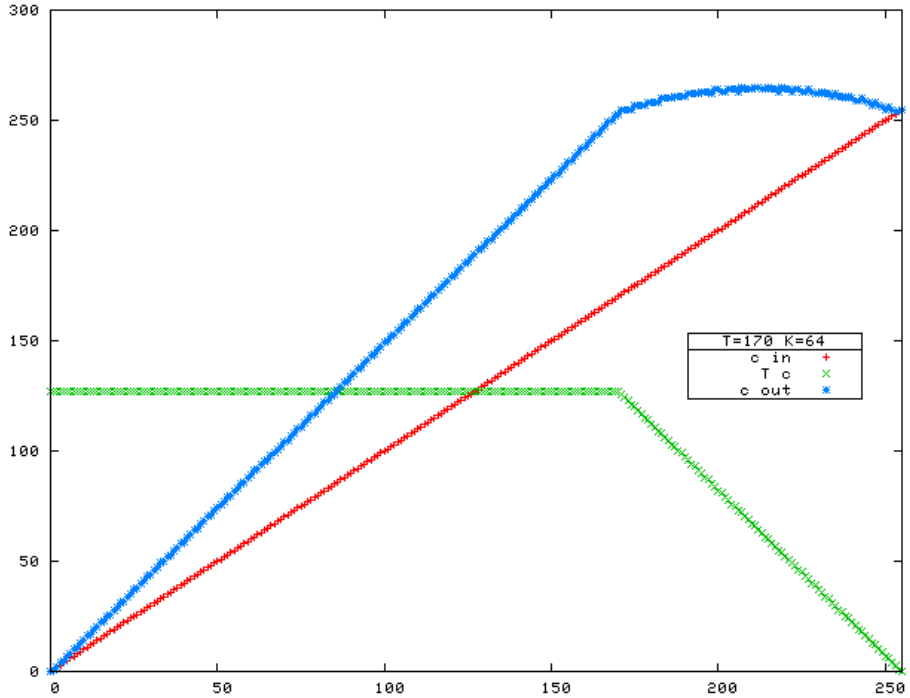
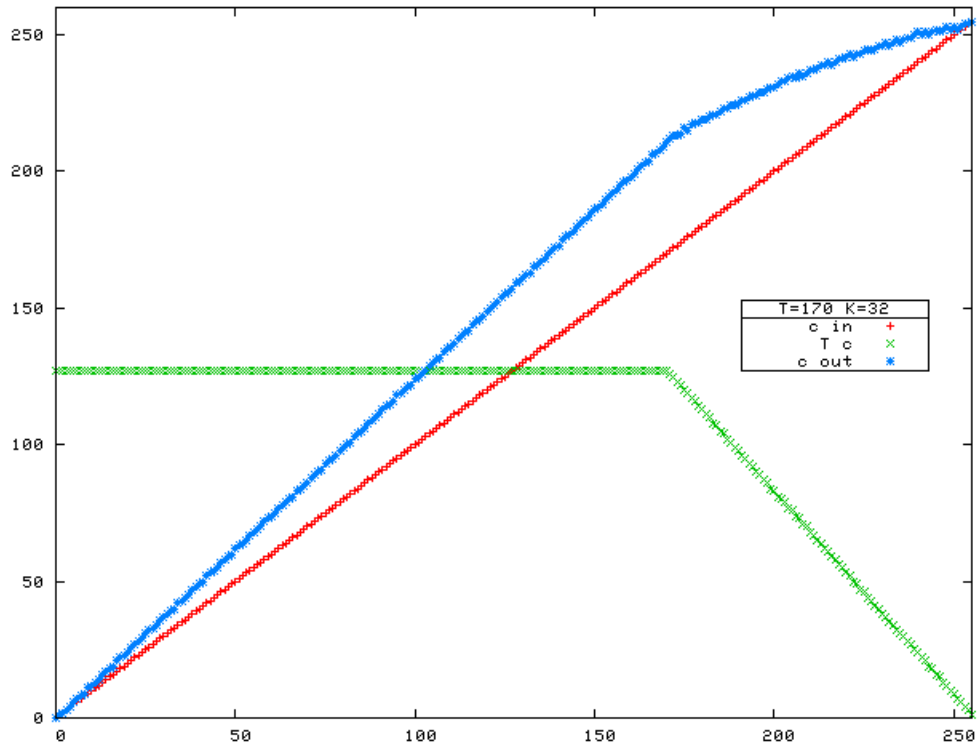


Figure 96: Threshold 170 at K=1.25



The programming interface consists of a control bit SOFT_CLIPPING_ENABLE and one register SD_SOFT_CLIPPING. The latter contains both the threshold and a software-computed reciprocal of inverse threshold. The software-computed reciprocal avoids the need to perform division in hardware. RECIP is computed as 64*1024 * 1.0 / (256 – threshold), truncated to 16-bit unsigned integer. Internally, the hardware may use fewer than 16 bits as described below.

When SOFT_CLIPPING_ENABLE=DISABLE, then SD_SOFT_CLIPPING is ignored and no clipping is done.

When SOFT_CLIPPING_ENABLE=ENABLE, SD_SOFT_CLIPPING is used. At the default values of SD_SOFT_CLIPPING, it behaves like Tegra 3.

The soft clipping equation performed in hardware, per component, is

```
If SD_CONTROL.SOFT_CLIPPING_ENABLE==ENABLE and c_in >= threshold:
T_c = 0x7f - (((c_in - threshold) * (recip >> 2)) >> 7);
Else
T_c = 0x7f
```

Where recip is the programmed SOFT_CLIPPING_RECIP; it is shifted right 2 bits to yield a 14-bit value. T_c is a 7-bit weight. The shift by 7 represents discarding 6 bits from recip plus one to give a 7-bit rather than 8-bit result. As before, T_c is used to scale the fractional portion of K:

```
c_k_limited = (K_c * T_c) >> 7;
```

and then apply the fractional gain to the component:

```
c_mul = (c_in * c_k_limited) >> 7;
```

and finally add the integer portion of gain (which is always 1, i.e., 1 * c_in = c_in):

```
c_out = c_in + c_mul
```

24.3.13.5 Increased Pixel Count Limits

With displays trending to 2560x1600 and larger panels, PRISM2 supports at least 2^{23} pixels. For simplification, the minimum screen size of 2^{16} is retained. Therefore the pixel count dynamic range increases from 32:1 to 128:1. The total_pixels and

histogram accumulator counters must support that range. Also, the histogram normalization must support an expanded range of exponents and shifting (0-7 rather than 0-5).

The SD_HISTOGRAM bins need not expand because they contain normalized pixel percentages, not raw counts.

SD_PIXEL_COUNT accommodates this range.

24.3.13.6 Run per-Frame on VPULSE2

For Tegra 3 devices and earlier, Smart Dimmer histogram processing was done on VSYNC, which is at the beginning of the frame. Thus a new BRIGHTNESS, based on the previous frame's histogram, was not available until the beginning of the next frame. In non-continuous (display self-refresh) mode, there can be a long delay. In Tegra 4 and later devices, PRISM2 can be configured to run on V_PULSE2 instead. This can be programmed to occur at the end of a frame, so that software can get the results in a timely manner, and decide whether to change brightness, re-transmit the frame with a new enhancement value, or neither.

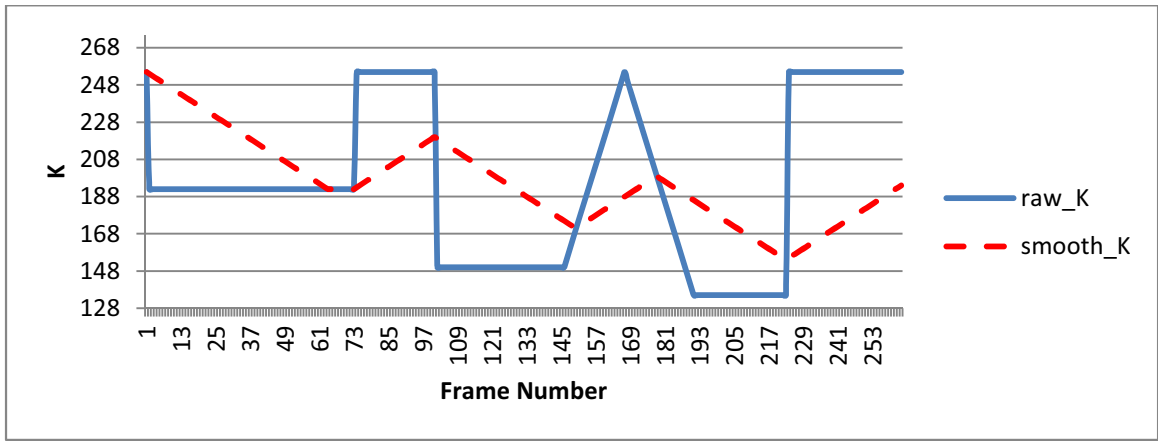
24.3.13.7 Programmable Smooth Transitions

Sudden changes in PRISM2 enhancement K and backlight can be noticeable and objectionable. While the soft-clipping compensation algorithm in Section 24.3.13.4: Programmable Soft Clipping above can help, it might not be enough. To improve the user experience, this feature allows changes to be automatically phased in (smoothed) over time.

When enabled, the hardware-generated raw K is the target value. It is not used directly for enhancement or backlight brightness. Rather, a "smooth_K" is used. Each frame, the current smooth_K is incremented or decremented if different than the target raw K. The smooth_K and increment value have four additional fraction bits. The minimum SMOOTH_K_INCR value of 0x1 changes smooth_K by 1/16 per frame. For a maximum raw_K change of 128, the maximum phase-in time is 128*16 = 2048 frames, or about 34 seconds.

The following figure illustrates the smooth_k function; SMOOTH_K_INCR = 64 which changes smooth_K by at most 1.0 per frame.

Figure 97: Smooth K



Software can compute SMOOTH_K_INCR as follows.

- Assume a maximum desired brightness change rate of R L* / second = SMOOTH_K_INCR K/frame.
- Assume L* is roughly linear with respect to K (L* = K/2).
- R L* / second = SMOOTH_K_INCR K/frame

Solving for SMOOTH_K_INCR. R L* / second = R L* / (60 frames) = R K / (2*60 frames) = R/120 K/frame.

So SMOOTH_K_INCR = R/120. Because the register is encoded as 8.6 fixed-point, encode as round (R/120 * 64).

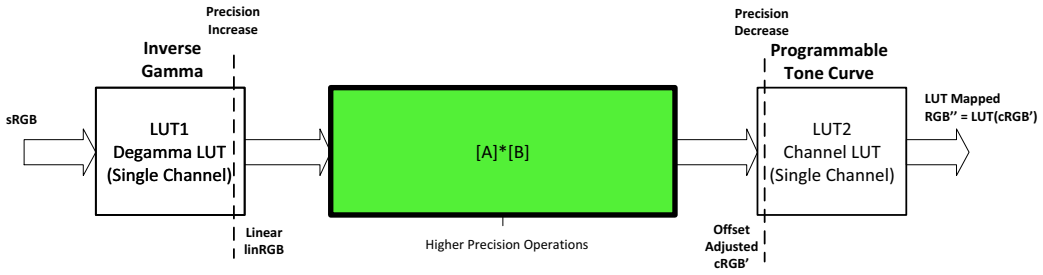
24.3.14 Display Color Management Unit (LUT/CSC)

This unit has an additional LUT -> CSC -> LUT after the PRISM2 unit to enable gamma and color gamut change. While this feature is independent from PRISM2, and more generally useful, it helps PRISM2 accommodate non-standard (non-sRGB) panels. It also helps convert from the display's native gamma and gamut (de facto of sRGB) into panels.

- First stage – LUT1 – is an 8-bit to 12-bit RAM-based look-up table (LUT) to convert gamma from sRGB 2.2 to linear (1.0). The output is 8.4 unsigned fixed-point format. A single LUT may be used for all 3 channels.
- Second stage – CSC – is 3x3 matrix multiply, to convert color gamut / whitepoint. 12 bits unsigned in, and 12 bits unsigned out. Its nine coefficients are 10-bit (S1.8 signed).
- Third stage – LUT2 – is a 12-bit to 8-bit LUT to convert from linear gamma to sRGB or similar gamma (for example, 1 / 2.2). A single LUT may be used for all 3 channels.

Note: *The window datapaths have subunits called LUT (CP) and CSC already, but they are for different purposes. The CP is for converting linear frame-buffer data into sRGB, or for color index modes, or for applying aesthetic changes like contrast. CSC is for converting YCbCr into RGB. They are for normalizing all surfaces into a common format, nominally sRGB, for blending.*

Figure 98: Color Management Unit



24.3.14.1 CMU LUT1

LUT1 maps 8-bit gamma-corrected input to 12-bit linear output. A single LUT may be used for all 3 RGB channels – per-channel control is not required.

```
R' = LUT1[R]
G' = LUT1[G]
B' = LUT1[B]
```

24.3.14.2 CSC Matrix

The CSC equation is:

$$\begin{bmatrix} KRR & KGR & KBR \\ KRG & KGG & KBG \\ KRB & KGB & KBB \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

24.3.14.3 CMU LUT2

LUT2 maps 12-bit linear input to 8-bit gamma corrected output. A single LUT may be used for all 3 RGB channels – per-channel control is not required. LUT2 is only 960 rows deep, and divided unevenly so that precision is variable; greater precision (more entries) are allocated to darker values. The first 512 entries 0..511, representing range [0.0, 32.0), are darker values, and are mapped with full 9-bit precision (5.4). The top 448 entries are for lower-precision brighter range [32, 256); they are reduced from 8.4 to 8.1 and offset so that values [32,256) map to 512..959 rather than 576..1023.

```
R' = LUT2[zone_select(R)]
```

```
G' = LUT2[zone_select(G)]
B' = LUT2[zone_select(B)]
```

The zone_select function maps a 12-bit component value to 10-bit RAM index, with variable precision:

```
If C >= THRESHOLD:
        C' = (C >> PRECISION_DIFF) + THRESHOLD – OVERLAP
Else:
        C' = C[9:0]
```

Where THRESHOLD=2^9 = 512, PRECISION_DIFF = 12 – 9 = 3, and OVERLAP = THRESHOLD >> PRECISION_DIFF = 512 >> 3 = 64.

Software would use an inverse_zone_select to initialize the LUT2.

24.3.14.4 Programming Interface

The CMU LUTs and CSC registers are single-buffered; shadow update and activation have no effect. However, the CMU_ENABLE bit is double-buffered, so it can be enabled/disabled at frame boundaries.

Host interface writes and reads take precedence over the datapath and may result in visible glitches. Software is responsible for ensuring the datapath is idle before writing or reading. An interrupt will be raised upon a conflict. The LUTs are guaranteed to be idle during blanking and when CMU_ENABLE=DISABLE. Software must explicitly enable reading, and reading is only performed for debugging reasons.

Note: *When reading LUTs: due to internal RAM latency, after changing read-enable or read address, data is not available until two clocks later. Best way to ensure this is to issue two back-to-back writes of same read address, as shown in read sequence.*

Software initialization sequence:

```
Ensure CMU_ENABLE=DISABLE or display idle
For C = 0 .. 255:
        C' = ungamma_function(C/255) * 4095
        CMU_LUT1.LUT1_ADDR = C
        CMU_LUT1.LUT1_DATA = C'
        Write CMU_LUT1
For C = 0 .. 1023: # assume 10-bit
        C' = gamma_function(inverse_zone_select(C)/4095) * 255
        CMU_LUT2.LUT1_ADDR = C
        CMU_LUT2.LUT1_DATA = C'
        Write CMU_LUT2
Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE
Use GENERAL_ACT_REQ
```

Where typically functions are: ungamma_function(x) = $x^{2.2}$, and gamma_function(x) = $x^{1/2.2}$.

LUT read sequence:

```
Ensure CMU_ENABLE=DISABLE or display idle
For C = 0 .. 255:
        CMU_LUT1_READ.LUT1_READ_EN = 1
        CMU_LUT1_READ.LUT1_READ_ADDR = C
        Write CMU_LUT1_READ
        Write CMU_LUT1_READ # ensure one cycle delay
        Read CMU_LUT1.LUT1_DATA
```

```
Write CMU_LUT1_READ.LUT1_READ_EN = 0
```

24.3.14.5 LUT Conflict Interrupt

To help debug conflicts caused by accessing LUTs while datapath is using them, CMU_LUT_CONFLICT interrupt is raised if:

- host write during scanout (pixel read when cmu_enable=1)
- host read during scanout
- host write during host read (i.e; LUT1_READ_EN=1)

24.3.15 Dither

Temporal dither is used in place of error diffusion and does not use line buffers.

24.3.16 Output RGB-to-YUV444 CSC

The Tegra X1 display outputs one of three color spaces:

- Red, Green, Blue color primaries.
- YCbCr 4:4:4 Luminance and color difference signals as defined by ITU-R BT.601
- YCbCr 4:4:4 Luminance and color difference signals as defined by ITU-R BT.709

YCbCr color space operates only on limited RGB color space of RGB[16, 235]. HDMI requires YCbCr support of limited color space and optionally extended color space. HDMI also allows for direct RGB limited color space input (that is, RGB[16, 235]). The Tegra X1 Display allows limitation of RGB color space automatically by placing hardcoded full to limited color space scaler unit prior to CSC. This allows for support of the following output formats to HDMI:

- RGB[0, 255]
- RGB[16, 235]
- BT.601 YCbCr 4:4:4 (converted from RGB[16, 235])
- BT.709 YCbCr 4:4:4 (converted from RGB[16, 235])

24.4 Programming Model

24.4.1 Software Alignment Requirements

For Tegra X1 devices, the pixel data alignment restriction is 64B boundaries. The table below describes what registers are affected.

Parameter	Register name	Pitch (packed) 1/ 2/4 Bpp	Pitch (Planar) 1 Bpp per surface	Tiled (packed) 1/2/4 Bpp	Tiled (planar) 1 Bpp per surface
line_stride	B_LINE_STRIDE	64	64	64	64
h_prescale_size	B_H_PRESCALED_SIZE	Bpp	Even	Bpp	Even
v_prescale_size	B_V_PRESCALED_SIZE	NA	Even for 420	NA	Even for 420
h_offset	B_ADDR_H_OFFSET	Bpp	Even	Bpp	Even
v_offset	B_ADDR_V_OFFSET	NA	Even for 420	NA	Even for 420
Startaddr	B_START_ADDR	64	64	256	256

24.4.2 DSC Programming Sequence

The Display Stream Compression (DSC) block is programmed only once on mode change, after that hardware just keeps running. There is no use case to change DSC configuration on the fly, and no hardware-software interaction needed when hardware is running.

Anytime software wants to change the mode, it has to stop the entire display pipeline. For DSC, either dsc auto_reset is enabled OR a 1 is written to soft_reset at frame end. On error (buffer over/underflow, timeout), software needs to reset the DSC module by writing 1 to soft_reset.

Table 127: Picture Parameter Set

	Register/(Field)	Value
dsc_version_major	NA	1
dsc_version_minor	NA	0
pps_identifier	NA	identifier that can be used to differentiate between different PPS tables
bits_per_component	NA	8
linebuf_depth	DC_COM_DSC_UNIT_SET_0 (DSC_LINEBUF_DEPTH)	Line buffer bit depth used to generate the stream, 8 or 9
block_pred_enable	DC_COM_DSC_COMMON_CTL_0 (DSC_BLOCK_PRED_ENABLE)	0 or 1
convert_rgb	NA	1
enable_422	NA	0
vbr_enable	NA	0
bits_per_pixel	DC_COM_DSC_COMMON_CTL_0 (DSC_BITS_PER_PIXEL)	bpp<<4, bpp is 8 , 12, or 16
pic_height	DSC_PIC_INFO Picture_height (V_DISP_ACTIVE)	Active picture height in lines
pic_width	DSC_PIC_INFO Picture_width (H_DISP_ACTIVE)	Active picture width in pixels
slice_height	DC_COM_DSC_SLICE_INFO_0 (DSC_SLICE_HEIGHT)	Slice height in pixel. When the whole picture is one slice, the slice height should equal to picture height.
slice_width	DC_COM_DSC_SLICE_INFO_0 (DSC_SLICE_WIDTH)	Slice width in pixel. When the whole picture is one slice, the slice width should equal to picture width.
chunk_size	DC_COM_DSC_COMMON_CTL_0 (DSC_CHUNK_SIZE)	ceil(bpp * slice_width / 8)
initial_xmit_delay	DC_COM_DSC_RC_DELAY_INFO_0 (DSC_INITIAL_XMIT_DELAY)	Int(4096/bpp)
initial_dec_delay	DC_COM_DSC_RC_DELAY_INFO_0 (DSC_INITIAL_DEC_DELAY)	minRateBufferSize = rc_model_size - initial_offset + ceil(initial_xmit_delay * bpp) + groupsPerLine * first_line_bpg_offset hrdDelay = ceil(minRateBufferSize /bpp) initial_dec_delay = hrdDelay - initial_xmit_delay
initial_scale_value	DC_COM_DSC_RC_SCALE_INFO_0 (DSC_INITIAL_SCALE_VALUE)	8*rc_model_size / (rc_model_size – initial_offset)

scale_increment_interval	DC_COM_DSC_RC_SCALE_INFO_2_0 (DSC_SCALE_INCR_INTERVAL)	final_scale= rc_model_size / (rc_model_size – final_offset)
		
		if (final_scale <= 9)
		scale_increment_interval = 0;
		else
		scale_increment_interval = groupsPerLine / (final_scale-9);

scale_decrement_interval	DC_COM_DSC_RC_SCALE_INFO_0 (DSC_SCALE_DECR_INTERVAL)	groupsPerLine / (initial_scale_value – 8)
first_line_bpg_offset	DC_COM_DSC_RC_FLATNESS_INFO_0 (DSC_FIRST_LINE_BPG_OFFS)	Suggested value:12 for 8bpp; 15 for 12bpp
nfl_bpg_offset	DC_COM_DSC_RC_BPGOFF_INFO_0 (DSC_NFL_BPG_OFFSET)	ceil((double)(first_line_bpg_ofs << 11) / (slice_height - 1));

Table 127: Picture Parameter Set

	Register/(Field)	Value
slice_bpg_offset	DC_COM_DSC_RC_BPGOFF_INFO_0 (DSC_SLICE_BPG_OFFSET)	ceil((double)(1<<11) * (rc_model_size - initial_offset + numExtraMuxBits)/ (groups_total)) numExtraMuxBits = 198 + ((chunk_size*slice_height*8-246)%48)
initial_offset	DC_COM_DSC_RC_OFFSET_INFO_0 (DSC_INITIAL_OFFSET)	Recommended value: 6144 for 8bpp; 2048 for 12bpp
final_offset	DC_COM_DSC_RC_OFFSET_INFO_0 (DSC_FINAL_OFFSET)	rc_model_size –initial_xmit_delay * bpp + numExtraMuxBits
flatness_min_qp	DC_COM_DSC_RC_FLATNESS_INFO_0 (DSC_FLATNESS_MIN_QP)	Recommended value: 3
flatness_max_qp	DC_COM_DSC_RC_FLATNESS_INFO_0 (DSC_FLATNESS_MAX_QP)	Recommended value:12
rc_model_size	DC_COM_DSC_RC_BUF_THRESH0_0 (DSC_RC_MODEL_SIZE)	Recommended value: 8192
rc_edge_factor	DC_COM_DSC_RC_PARAM_SET_0 (DSC_RC_EDGE_FACTOR)	Recommended value: 6
rc_quant_incr_limit0	DC_COM_DSC_RC_PARAM_SET_0 (DSC_RC_QUANT_INCR_LIMIT0)	Recommended value: 11
rc_quant_incr_limit1	DC_COM_DSC_RC_PARAM_SET_0 (DSC_RC_QUANT_INCR_LIMIT1)	Recommended value: 11
rc_tgt_offset_hi	DC_COM_DSC_RC_PARAM_SET_0 (DSC_RC_TGT_OFFSET_HI)	Recommended value: 3
rc_tgt_offset_lo	DC_COM_DSC_RC_PARAM_SET_0 /Rc_tgt_offset_lo	Recommended value: 3
rc_buf_thresh[14]	DC_COM_DSC_RC_BUF_THRESH[0~3]_0	Recommended value: 896, 1792, 2688, 3584, 4480, 5376, 6272, 6720, 7168,7616, 7744, 7872, 8000, 8064(the top most three thresh need change to 7808, 7872, 7936 for rc overflow solution 3)
rc_range_parameters[15]	DC_COM_DSC_RC_RANGE_CFG[0~7]_0	Recommended value: see spec table 16-2
Other info:		
output_delay	DC_COM_DSC_DELAY_0 (DSC_OUTPUT_DELAY)	delay_in_slice = (0x17f*8*3+bpp-1)/bpp + slice_width+ initial_xmit_delay+18 (delay_in_slice/slice_width)*(pic_width+hBlank) + (delay_in_slice%slice_width)
timeout_counter	DC_COM_DSC_TOP_CTL_0 (DSC_TIMEOUT_COUNTER)	It can be set to slightly larger than HBLANK 0 means disable
intr_mask	DSC_INTR /intr_mask (INT_MASK.DSC_*_INT_MASK registers)	1: mask, 0: not mask
hblank_cycles	Reuse display	Hblank cycles
Slice_num_minus1_in_line	DC_COM_DSC_UNIT_SET_0 (DSC_SLICE_NUM_MINUS1_IN_LINE)	0, 1 or 3
Soft_reset	DC_COM_DSC_TOP_CTL_0 (DSC_SOFT_RESET)	0: disable, 1: enable
Auto_reset	DC_COM_DSC_TOP_CTL_0 (DSC_AUTO_RESET)	0: disable, 1:enable
Slcg_override	DC_COM_DSC_TOP_CTL_0 (DSC_SLCG_OVERRIDE)	0:disable, 1:enable

Table 127: Picture Parameter Set

	Register/(Field)	Value
rc_solution_mode	DC_DISP_DISPLAY_SPARE0_0 (rc_solution_mode)	Select RC solution for overflow issue: 0 disable 1: solution 2 2: solution 3
check_flatness2	DC_DISP_DISPLAY_SPARE0_0 (check_flatness2)	Enable flatness check step 2 for overflow solution
flatness_fix_en	DC_DISP_DISPLAY_SPARE0_0 (flatness_fix_en)	Enable flatness fix for overflow solutions
rc_overflow_thresh	DC_DISP_DISPLAY_SPARE0_0 (rc_overflow_thresh)	RC threshold for overflow solution 2, range can be -512~512

24.5 Display Controller Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers. The display memory map is shown below.

24.5.1 Address Map

Space Name	Offset	Notes
DC_CMD	0x0	Non-shadowed command/sync registers
DC_D_WIN	0x80	Window D instance of DC_WIN
DC_D_WINBUF	0xc0	Window D instance of DC_WINBUF
DC_T_WIN	0x100	Window T instance of DC_WIN
DC_T_WINBUF	0x140	Window T instance of DC_WINBUF
DC_H_WIN	0x180	Window H instance of DC_WIN (RESERVED)
DC_H_WINBUF	0x1c0	Window H instance of DC_WINBUF (RESERVED)
DC_COM	0x300	Non-shadowed non-window registers
DC_DISP	0x400	Shadowed non-window registers
DC_WINC	0x500	Non-shadowed indirect window registers: indirect alias; windows A/B/C; use DISPLAY_WINDOW_HEADER(1)
DC_WIN	0x700	Shadowed indirect window register: indirect alias; windows A/B/C/D; use DISPLAY_WINDOW_HEADER(1)
DC_WINBUF	0x800	Triple-buffered indirect window registers: indirect alias; windows A/B/C/D; use DISPLAY_WINDOW_HEADER(1)
DC_A_WINC	0xa00	Window A instance for DC_WINC
DC_A_WIN	0xb80	Window A instance for DC_WIN
DC_A_WINBUF	0xbc0	Window A instance for DC_WINBUF
DC_B_WINC	0xc00	Window B instance for DC_WINC
DC_B_WIN	0xd80	Window B instance for DC_WIN
DC_B_WINBUF	0xdc0	Window B instance for DC_WINBUF
DC_C_WINC	0xe00	Window C instance for DC_WINC
DC_C_WIN	0xf80	Window C instance for DC_WIN
DC_C_WINBUF	0xfc0	Window C instance for DC_WINBUF
DC_D_WINC	0x1000	Window D instance of DC_WINC (RESERVED)
DC_T_WINC	0x1200	Window T instance of DC_WINC (RESERVED)
DC_H_WINC	0x1400	Window H instance of DC_WINC (RESERVED)
Note the indirect ranges have been deprecated but are listed for legacy purposes.		

24.5.2 New Registers for Tegra X1 Devices

The following registers or register fields are new or changed for Tegra X1 devices.

- DISP_WIN_OPTIONS. SOR1_ENABLE – similar to SOR_ENABLE. Enables output of pixels from this controller to SOR1. Note that SOR1 has a separate input selector register.
- SECURE_CONTROL. SECURE_SOR1_PROTECT - In TrustZone mode, this allows TrustZone code to disable the output to SOR1.
- Windows A/B/C CDE registers
- DSC registers and register fields

24.5.3 Display Controller Register Definition

Each display supports three windows: Window A, Window B, Window C

The windows may have different features, as described above; however, the register interfaces are the same.

Color Palette

These appear as three instances of 256 32-bit registers with each register consisting of one RGB pixel so not all 32 bits in the registers are used. One instance is used for window A, another instance is used for window B, and the third instance is used for window C. In reality, each instance is implemented as triple 256-word dual-port register files (2P RF) with one read port and one write port (1R1W) and with each word consisting of 1 red/green/blue component. Read port of the color palettes are used for the corresponding window A, window B, or window C and write port of the color palettes are used for host write.

Note that the host cannot read these color palettes, so it must cache this color palette somewhere else to be able to read them. This is done to reduce area.

24.5.4 Display Shadow Registers

Display registers have three shadow types:

1. Not Shadowed

   Writes to these registers take effect immediately.

2. Double Buffered

   Each register has two copies: ASSEMBLY and ACTIVE. ACTIVE is the working copy.

   These two copies share the same address/offset.

   WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:

   - When set to ACTIVE, both ACTIVE and ASSEMBLY copies will be written
   - When set to ASSEMBLY, only ASSEMBLY copy will be written

   READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

   If display is in STOP mode, ASSEMBLY copy is latched into ACTIVE copy immediately after

   (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

3. Triple Buffered

   Each register has three copies: ASSEMBLY, ARM, and ACTIVE. ACTIVE is the working copy.

   ASSEMBLY and ACTIVE copies share the same address/offset, the ARM copy is located at the address/offset one bigger than the ASSEMBLY/ACTIVE copy.

   WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:

   - When set to ACTIVE, all three copies will be written
   - When set to ASSEMBLY, only ASSEMBLY copy will be written

READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

- To read back ASSEMBLY or ACTIVE copy, set READ_MUX correctly before register reads.

These two copies share the same address.

- To read back ARM copy, do not set READ_MUX, but read back the register/register-field with "_NS" in their names, the offset of which is 1 bigger than its ASSEMBLY/ACTIVE counterpart.

ASSEMBLY copy is latched into ARM copy immediately after GENERAL/WIN_A/WIN_B/WIN_C)_UPDATE is programmed.

If display is in STOP mode, ARM copy is latched into ACTIVE copy immediately after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

24.5.5 Display Control Modes

The DISPLAY_COMMAND is used to set display control mode (**CONTINUOUS**, **ONE-SHOT**, or **STOP**).

Display switches mode when CONTROL_MODE is programmed to shadow register and then activated.

Display enters a mode when the register activation happens, either on frame boundary when entering **STOP** mode, or immediately when exiting **STOP** mode.

In **STOP** mode, display is in idle. Pixel clock is not running.

In **CONTINUOUS** mode, display keeps refreshing the output frame by frame. This mode is mostly used for the panels without internal frame buffer.

In **ONE-SHOT** mode, also known as non-continuous mode, display waits for a trigger before refreshing each frame. Between those frames, display is in idle. This mode is usually for the panels with internal frame buffer.

In ONE-SHOT mode, there are different types of triggers as follows:

- Input triggers

  Input trigger is to notify DISPLAY that a new memory surface is ready to be displayed.

  - Host Trigger

  Host trigger is requested when NC_HOST_TRIG_ENABLE is programmed. This register field is in the same register as shadow registers UPDATE/ACT_REQ so that the trigger can happen atomically with register updates for the new frame. When host trigger is requested within a frame, the requested frame will start immediately after the end of the current frame.

- Output Triggers

  Output trigger is to notify DISPLAY that a panel is ready to receive data from DISPLAY.

  - Tear Effect Signal Triggers

  When MSF/SSF register field is set to ENABLE, DISPLAY holds off a frame requested by input triggers until DISPLAY receives a "ready" from the pins that connect to the tearing effect signals from the panel, then sends a new frame to panel. However, if no input trigger has been requested, DISPLAY does not send a new frame.

In CONTINUOUS mode, the meaning of the triggers is different.

- Input Triggers

  Since in CONTINUOUS mode trigger happen automatically and repeatedly, trigger here only affects the buffer address change on a window.

  - Host Trigger

    Host can change the buffer address and then write WIN_A/B/C_ACT_REQ to request the buffer address change. Note that for syncpt logic, it cannot tell if a WIN_A/B/C register activation indicates the change of address, so it always behaves like a change of address happens on any register activation requested by WIN_A/B/C_ACT_REQ. As the result, RD_DONE/OP_DONE is returned.

- Output triggers

  These triggers are not applicable in CONTINUOUS mode. MSF/SSF_ENABLE should never be set in CONTINUOUS mode.

24.5.6 Sync Points

DISPLAY has 4 syncpt clients:

- GENERAL

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE not meaningful, same as IMMEDIATE
  - 2: RD_DONE not meaningful, same as IMMEDIATE
  - 3: REG_WR_SAFE returns INDX whenever it is safe to program GENERAL shadow registers (when all previous GENERAL activation has happened)
  - 4: HSPI returns INDX whenever it is safe to send HSPI data (when all HSPI_*** registers are safe to be programmed)
  - 5: FRAME_DONE returns INDX on the next frame end
  - 6: VPULSE3 returns INDX on the next vpulse3 leading edge (for timing sensitive operations if needed)
  - 7: FRAME_START returns INDX on the next frame start (currently for hardware testing, but can be used if needed)

- WIN_A

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE same as RD_DONE
  - 2: RD_DONE returns INDX when all WIN_A reads for frames activated before the INCR_SYNCPT are complete. In ONE-SHOT mode with TVO disabled, this happens after the last row of window A display. In CONTINUOUS mode, or in ONE-SHOT mode with TVO enabled, this happens on the last row of window A or on frame end (depending if the INCR_SYNCPT is issued before or after the window A last row).
  - 3: REG_WR_SAFE returns INDX whenever it is safe to program WIN_A shadow registers (when all previous WIN_A activation has happened)
  - 4: LINE_FLIPPED returns INDX whenever the activation has happened and the previous surface has been completely read out. This is particularly useful for line flip case, when s/w could use the syncpt to free up the previously read surface. As hardware moves on to the new surface.

- WIN_B

  Similar to WIN_A

- WIN_C

  Similar to WIN_A

- DISPLAY Continuous syncpt

  VSYNC: When enabled, return syncpt whenever a vblank start happens.

24.5.7 Raise Actions (Legacy)

The Raise action is used to check the state of the display's command execution. The Raise action can be enabled by itself or with other actions.

The raise value is returned to the host controller if the current and all previous commands have completed execution. If the command is executed with the delayed mode, the raise is not returned till that command is executed at the end of the frame.

If there is no pending command when the Raise is issued, the raise value is returned immediately. The raise value is a 4-bit number included in the command.

Context switch acknowledge, register reads, and raises are returned to the host through a read FIFO. However, because there may be more than one raise, read, or acknowledge available for writing to the FIFO in a certain cycle, there must be a priority encoder. Here is the priority:

1. Context switch
2. Register read
3. SIGNAL_RAISE
4. SIGNAL_RAISE1
5. SIGNAL_RAISE2
6. SIGNAL_RAISE3
7. DISP_COMMAND_RAISE
8. HSPI_RAISE
9. Refcount

These packets are for verification; they have no relevance to software. The RAISE packet should include the intended return channel. This channel should be passed back to the host so the RAISE will be reflected in the correct channel.

24.6 Display CMD Registers

24.6.1 DC_CMD_GENERAL_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	GENERAL_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = HSPI 5 = FRAME_DONE 6 = VPULSE3 7 = FRAME_START 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	GENERAL_INDX: syncpt index value

24.6.2 DC_CMD_GENERAL_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	GENERAL_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	GENERAL_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

24.6.3 DC_CMD_GENERAL_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GENERAL_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

24.6.4 DC_CMD_WIN_A_INCR_SYNCPT_0

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	WIN_A_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = LINE_FLIPPED 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_A_INDX: syncpt index value

24.6.5 DC_CMD_WIN_A_INCR_SYNCPT_CNTRL_0

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_A_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_A_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

24.6.6 DC_CMD_WIN_A_INCR_SYNCPT_ERROR_0

Offset: 0xa | Byte Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_A_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

24.6.7 DC_CMD_WIN_B_INCR_SYNCPT_0

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	WIN_B_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = LINE_FLIPPED 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_B_INDX: syncpt index value

24.6.8 DC_CMD_WIN_B_INCR_SYNCPT_CNTRL_0

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	WIN_B_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.

Bit	Reset	Description
0	0x0	WIN_B_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

24.6.9 DC_CMD_WIN_B_INCR_SYNCPT_ERROR_0

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_B_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

24.6.10 DC_CMD_WIN_C_INCR_SYNCPT_0

Offset: 0x18 | Byte Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	WIN_C_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = LINE_FLIPPED 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_C_INDX: syncpt index value

24.6.11 DC_CMD_WIN_C_INCR_SYNCPT_CNTRL_0

Offset: 0x19 | Byte Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	WIN_C_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_C_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

24.6.12 DC_CMD_WIN_C_INCR_SYNCPT_ERROR_0

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_C_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

24.6.13 DC_CMD_CONT_SYNCPT_VSYNC_0

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxxx)

Bit	Reset	Description
8	0x0	VSYNC_EN: On host read bus every time VSYNC (V-blank leading edge) happens and VSYNC_EN is set 0 = DISABLE 1 = ENABLE
7:0	X	VSYNC_INDX: Return INDX (set HOST_CLRD packet TYPE field to SYNCPT)

24.6.14 DC_CMD_CTXSW_0

Context switch registers for class and channel

Should be common to all modules. Includes the current channel/class (writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit.

This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x30 | Byte Offset: 0xc0 | Read/Write: R/W | Reset: 0xf000f800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

24.6.15 DC_CMD_DISPLAY_COMMAND_OPTION0_0

Display Controller Option 0. This register is not effective until DISPLAY_COMMAND is written.

Class: Display Command

Offset: 0x31 | Byte Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx000xxxxxxx00000000)

Bit	Reset	Description
18	0x0	WINDOW_C_NC_DISPLAY: Window C Non-Continuous Display. This is effective only in Non-Continuous Display mode when window C buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window C buffer is switched. 0 = DISABLE 1 = ENABLE
17	0x0	WINDOW_B_NC_DISPLAY: Window B Non-Continuous Display. This is effective only in Non-Continuous Display mode when window B buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window B buffer is switched. 0 = DISABLE 1 = ENABLE
16	0x0	WINDOW_A_NC_DISPLAY: Window A Non-Continuous Display. This is effective only in Non-Continuous Display mode when window A buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window A buffer is switched. 0 = DISABLE 1 = ENABLE
7:6	0x0	SSF_SOURCE: Source pin for the SSF input. Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflect this historical mapping. 0= LCD_DC pin (legacy default) 1= LCD_SPI pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = SSF_LDC 1 = SSF_LSPI 2 = SSF_LSDI
5	0x0	SSF_ENABLE: Sub-Display Stop Frame (SSF) input. This is effective only in Non-Continuous Display mode. When enabled, SSF signal can be input through LDC pin. When SSF is enabled a trigger to send a frame in Non-Continuous Display mode will be delayed until SSF is active. 0 = DISABLE 1 = ENABLE
4	0x0	SSF_POLARITY: Sub-Display Stop Frame (SSF) Polarity 0= Active high 1= Active low
3:2	0x0	MSF_SOURCE: Source pin for the MSF input. Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflects this historical mapping. 0= LCD_SPI pin (legacy default) 1= LCD_DC pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = MSF_LSPI 1 = MSF_LDC 2 = MSF_LSDI
1	0x0	MSF_ENABLE: Main-Display Stop Frame (MSF) input This is effective only in Non-Continuous Display mode. When enabled, the MSF signal can be input through the LSPI pin. When MSF is enabled, a trigger to send a frame in Non-Continuous Display mode will be delayed until MSF is active. 0 = DISABLE 1 = ENABLE
0	0x0	MSF_POLARITY: Main-Display Stop Frame (MSF) Polarity 0= Active high 1= Active low

24.6.16 DC_CMD_DISPLAY_COMMAND_0

Display Command

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0)

Bit	Reset	Description
30:27	X	DISP_COMMAND_RAISE_CHANNEL_ID: Display Command Channel ID
26:22	X	DISP_COMMAND_RAISE_VECTOR: Display Command Raise Vector. This raise vector is at the next line or frame boundary, depending on GENERAL_ACT_CNTR_SEL

Bit	Reset	Description
6:5	0x0	DISPLAY_CTRL_MODE: Display Controller Mode 0= Stop Display, this can be used to stop sending frame at the next frame boundary. This is automatically generated in Non-Continuous Display after sending one frame. If this is issued when display controller is already stopped then there is no frame sent. Raise vector (if raise is enabled) is also returned immediately. This command can also be used in non-continuous display mode to stop accepting non-host trigger conditions from other clients. 1= Continuous Display, the display controller will continuously send frame. Continuous display mode can be stopped by switching to Non-Continuous Display or by issuing Stop Display. 2= Non-Continuous Display, the display controller is forced to send one frame of each active display and then wait for the next time this command is issued or for other (non-host) trigger conditions to send frame. The sending of frames may be delayed by MSF or SSF input signals from the display device. If a Stop Display is issued while in non-continuous display mode then non-host trigger conditions will no longer be accepted until the next time Non-Continuous Display is issued 0 = STOP 1 = C_DISPLAY 2 = NC_DISPLAY
0	0x0	DISP_COMMAND_RAISE: Display Command Raise. Raise vector will be returned at the end of command completion 0 = DISABLE 1 = ENABLE

24.6.17 DC_CMD_SIGNAL_RAISE_0

When written, the next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE_TYPE option so that multiple raises can be returned without software intervention. SIGNAL_RAISE1, SIGNAL_RAISE2, or SIGNAL_RAISE3 can be used if more than one source signal is needed.

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE_SELECT=NONE or SIGNAL_RAISE_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE_VECTOR: bit number to raise

24.6.18 DC_CMD_DISPLAY_POWER_CONTROL_0

PW0, PW1, PW2, and PW3 are tied internally to the same value corresponding to PW3.

Software is required to program all these register fields to ENABLE - PW0 to PW3 simultaneously, in the same register write access.

The power sequencing provided by PW0 - PW3 was LCD output specific, which is not applicable because LCD has been removed.

Display Power Control

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00xxxxx0x0xxxxxxx0x0x0x0x0)

Bit	Reset	Description
25	0x0	HSPI_ENABLE: Host SPI write cycle enable. SPI_ENABLE must be enabled also for this bit to be effective. 0 = DISABLE 1 = ENABLE
24	0x0	SPI_ENABLE: SPI interface enable. This enables clock to SPI interface logic for Host SPI, IS SPI, and LCD SPI. 0 = DISABLE 1 = ENABLE
18	0x0	PM1_ENABLE: PM1 signal enable 0 = DISABLE 1 = ENABLE
16	0x0	PM0_ENABLE: PM0 signal enable 0 = DISABLE 1 = ENABLE
8	0x0	PW4_ENABLE: PW4 signal enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
6	0x0	PW3_ENABLE: PW3 signal enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
4	0x0	PW2_ENABLE: PW2 signal enable. This signal controls pixel data processing. It should be enabled during V blank time. This signal also controls the time when pin polarity takes effect at the pad. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
2	0x0	PW1_ENABLE: PW1 signal enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
0	0x0	PW0_ENABLE: PW0 signal enable. This signal controls the display H and V counters. It must be enabled first and disabled last during display power sequencing. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE

24.6.19 DC_CMD_INT_STATUS_0

This reflects status of all pending interrupts which is valid as long as the interrupt is not cleared even if the interrupt is masked. A pending interrupt can be cleared by writing a '1' to this the corresponding interrupt status bit in this register.

Display Interrupt and Status

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	DSC_TO_UF_INT: DSC TO Underflow Interrupt Status. 1 = Interrupt pending 0 = Interrupt not pending
28	X	DSC_BBUF_UF_INT: DSC BBUF Underflow Interrupt Status. 1 = Interrupt pending 0 = Interrupt not pending

Bit	Reset	Description
27	X	DSC_RBUF_UF_INT: DSC RBUF Underflow Interrupt Status. 1 = Interrupt pending 0 = Interrupt not pending
26	X	DSC_OBUF_UF_INT: DSC OBUF Underflow Interrupt Status. 1 = Interrupt pending 0 = Interrupt not pending
25	X	WIN_T_UF_INT: Window T Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
24	X	WIN_D_UF_INT: Window D Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
23	X	HC_UF_INT: Cursor Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
22	X	CMU_LUT_CONFLICT_INT: CMU LUT read/write conflict; write 1 to clear
16	X	WIN_C_OF_INT: Window C Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
15	X	WIN_B_OF_INT: Window B Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
14	X	WIN_A_OF_INT: Window A Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
13	X	SSF_INT: Sub-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
12	X	MSF_INT: Main-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
10	X	WIN_C_UF_INT: Window C Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
9	X	WIN_B_UF_INT: Window B Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
8	X	WIN_A_UF_INT: Window A Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
7	X	SPI_BUSY_INT: SPI Busy Interrupt Status 0= interrupt not pending 1= interrupt pending
5	X	V_PULSE2_INT: Vertical Pulse 2 Interrupt 0= interrupt not pending 1= interrupt pending
4	X	V_PULSE3_INT: Vertical Pulse 3 Interrupt 0= interrupt not pending 1= interrupt pending
3	X	H_BLANK_INT: Horizontal Blank Interrupt 0= interrupt not pending 1= interrupt pending
2	X	V_BLANK_INT: Vertical Blank Interrupt 0= interrupt not pending 1= interrupt pending

Bit	Reset	Description
1	X	FRAME_END_INT: Frame End Interrupt 0= interrupt not pending 1= interrupt pending
0	X	CTXSW_INT: Context Switch Interrupt Status (this is cleared on write) 0= interrupt not pending 1= interrupt pending

24.6.20 DC_CMD_INT_MASK_0

Setting bits in this register masks the corresponding interrupt but neither clears a pending interrupt nor prevents a pending interrupt from being generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status from being generated.

Interrupt Mask

Offset: 0x38 | Byte Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000xxxxx00000x0000x000000)

Bit	Reset	Description
29	0x0	DSC_TO_UF_INT_MASK: DSC TO Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
28	0x0	DSC_BBUF_UF_INT_MASK: DSC BBUF Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
27	0x0	DSC_RBUF_UF_INT_MASK: DSC RBUF Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
26	0x0	DSC_OBUF_UF_INT_MASK: DSC OBUF Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
25	0x0	WIN_T_UF_INT_MASK: Window T Underflow Interrupt Mask. 0 = MASKED 1 = NOTMASKED
24	0x0	WIN_D_UF_INT_MASK: Window D Underflow Interrupt Mask. 0 = MASKED 1 = NOTMASKED
23	0x0	HC_UF_INT_MASK: Cursor Underflow Interrupt Mask. 0 = MASKED 1 = NOTMASKED
22	0x0	CMU_LUT_CONFLICT_INT_MASK: CMU LUT read/write conflict. 0 = MASKED 1 = NOTMASKED
16	0x0	WIN_C_OF_INT_MASK: Window C Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
15	0x0	WIN_B_OF_INT_MASK: Window B Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
14	0x0	WIN_A_OF_INT_MASK: Window A Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
13	0x0	SSF_INT_MASK: Sub-Display Stop Frame Interrupt Mask 0 = MASKED 1 = NOTMASKED
12	0x0	MSF_INT_MASK: Main-Display Stop Frame Interrupt Mask 0 = MASKED 1 = NOTMASKED
10	0x0	WIN_C_UF_INT_MASK: Window C Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
9	0x0	WIN_B_UF_INT_MASK: Window B Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
8	0x0	WIN_A_UF_INT_MASK: Window A Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
7	0x0	SPI_BUSY_INT_MASK: SPI Busy Interrupt Mask 0 = MASKED 1 = NOTMASKED
5	0x0	V_PULSE2_INT_MASK: Vertical Pulse 2 Interrupt Mask 0 = MASKED 1 = NOTMASKED
4	0x0	V_PULSE3_INT_MASK: Vertical Pulse 3 Interrupt Mask 0 = MASKED 1 = NOTMASKED
3	0x0	H_BLANK_INT_MASK: Horizontal Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
2	0x0	V_BLANK_INT_MASK: Vertical Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
1	0x0	FRAME_END_INT_MASK: Frame End Interrupt Mask 0 = MASKED 1 = NOTMASKED
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED 1 = NOTMASKED

24.6.21 DC_CMD_INT_ENABLE_0

Setting bits in this register enable the corresponding interrupt event to generate a pending interrupt. Interrupt output signal will be activated only if the corresponding interrupt is not masked.

Disabling an interrupt will not clear a corresponding pending interrupt - it only prevents a new interrupt event to generate a pending interrupt.

Interrupt Enable

Offset: 0x39 | Byte Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000001 (0bxx00000000xxxxx00000x0000x000001)

Bit	Reset	Description
29	0x0	DSC_TO_UF_INT_ENABLE: DSC TO Underflow Interrupt Enable. 1 = Interrupt enabled 0 = Interrupt disabled 0 = DISABLE 1 = ENABLE
28	0x0	DSC_BBUF_UF_INT_ENABLE: DSC BBUF Underflow Interrupt Enable. 1 = Interrupt enabled 0 = Interrupt disabled 0 = DISABLE 1 = ENABLE
27	0x0	DSC_RBUF_UF_INT_ENABLE: DSC RBUF Underflow Interrupt Enable. 1 = Interrupt enabled 0 = Interrupt disabled 0 = DISABLE 1 = ENABLE
26	0x0	DSC_OBUF_UF_INT_ENABLE: DSC OBUF Underflow Interrupt Enable. 1 = Interrupt enabled 0 = Interrupt disabled 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x0	WIN_T_UF_INT_ENABLE: Window T Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
24	0x0	WIN_D_UF_INT_ENABLE: Window D Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
23	0x0	HC_UF_INT_ENABLE: Cursor Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
22	0x0	CMU_LUT_CONFLICT_INT_ENABLE: CMU LUT read/write conflict 0 = DISABLE 1 = ENABLE
16	0x0	WIN_C_OF_INT_ENABLE: Window C Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
15	0x0	WIN_B_OF_INT_ENABLE: Window B Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
14	0x0	WIN_A_OF_INT_ENABLE: Window A Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
13	0x0	SSF_INT_ENABLE: Sub-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
12	0x0	MSF_INT_ENABLE: Main-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
10	0x0	WIN_C_UF_INT_ENABLE: Window C Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
9	0x0	WIN_B_UF_INT_ENABLE: Window B Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
8	0x0	WIN_A_UF_INT_ENABLE: Window A Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
7	0x0	SPI_BUSY_INT_ENABLE: SPI Busy Interrupt Enable 0 = DISABLE 1 = ENABLE
5	0x0	V_PULSE2_INT_ENABLE: Vertical Pulse 2 Interrupt Enable 0 = DISABLE 1 = ENABLE
4	0x0	V_PULSE3_INT_ENABLE: Vertical Pulse 3 Interrupt Enable 0 = DISABLE 1 = ENABLE
3	0x0	H_BLANK_INT_ENABLE: Horizontal Blank Interrupt Enable 0 = DISABLE 1 = ENABLE
2	0x0	V_BLANK_INT_ENABLE: Vertical Blank Interrupt Enable 0 = DISABLE 1 = ENABLE
1	0x0	FRAME_END_INT_ENABLE: Frame End Interrupt Enable 0 = DISABLE 1 = ENABLE
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE 1 = ENABLE

24.6.22 DC_CMD_INT_TYPE_0

Two interrupt types are available:

- Edge interrupt - transition on input signal/event generates pending interrupt
- Level interrupt - active level on input signal/event generates pending interrupt

Interrupt Type

Offset: 0x3a | Byte Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000xxxxx00000x0000x00000x)

Bit	Reset	Description
29	0x0	DSC_TO_UF_INT_TYPE: DSC TO Underflow Interrupt Type. 1 = Level Interrupt 0 = Edge Interrupt 0 = EDGE 1 = LEVEL
28	0x0	DSC_BBUF_UF_INT_TYPE: DSC BBUF Underflow Interrupt Type. 1 = Level Interrupt 0 = Edge Interrupt 0 = EDGE 1 = LEVEL
27	0x0	DSC_RBUF_UF_INT_TYPE: DSC RBUF Underflow Interrupt Type. 1 = Level Interrupt 0 = Edge Interrupt 0 = EDGE 1 = LEVEL
26	0x0	DSC_OBUF_UF_INT_TYPE: DSC OBUF Underflow Interrupt Type. 1 = Level Interrupt 0 = Edge Interrupt 0 = EDGE 1 = LEVEL
25	0x0	WIN_T_UF_INT_TYPE: Window T Underflow Interrupt Type 0 = EDGE 1 = LEVEL
24	0x0	WIN_D_UF_INT_TYPE: Window D Underflow Interrupt Type 0 = EDGE 1 = LEVEL
23	0x0	HC_UF_INT_TYPE: Cursor Underflow Interrupt Type 0 = EDGE 1 = LEVEL
22	0x0	CMU_LUT_CONFLICT_INT_TYPE: CMU LUT read/write conflict. Must be EDGE. 0 = EDGE 1 = LEVEL
16	0x0	WIN_C_OF_INT_TYPE: Window C Overflow Interrupt Type 0 = EDGE 1 = LEVEL
15	0x0	WIN_B_OF_INT_TYPE: Window B Overflow Interrupt Type 0 = EDGE 1 = LEVEL
14	0x0	WIN_A_OF_INT_TYPE: Window A Overflow Interrupt Type 0 = EDGE 1 = LEVEL
13	0x0	SSF_INT_TYPE: Sub-Display Stop Frame Interrupt Type 0 = EDGE 1 = LEVEL
12	0x0	MSF_INT_TYPE: Main-Display Stop Frame Interrupt Type 0 = EDGE 1 = LEVEL
10	0x0	WIN_C_UF_INT_TYPE: Window C Underflow Interrupt Type 0 = EDGE 1 = LEVEL
9	0x0	WIN_B_UF_INT_TYPE: Window B Underflow Interrupt Type 0 = EDGE 1 = LEVEL

Bit	Reset	Description
8	0x0	WIN_A_UF_INT_TYPE: Window A Underflow Interrupt Type 0 = EDGE 1 = LEVEL
7	0x0	SPI_BUSY_INT_TYPE: SPI Busy Interrupt Type 0 = EDGE 1 = LEVEL
5	0x0	V_PULSE2_INT_TYPE: Vertical Pulse 2 Interrupt Type 0 = EDGE 1 = LEVEL
4	0x0	V_PULSE3_INT_TYPE: Vertical Pulse 3 Interrupt Type 0 = EDGE 1 = LEVEL
3	0x0	H_BLANK_INT_TYPE: Horizontal Blank Interrupt Type 0 = EDGE 1 = LEVEL
2	0x0	V_BLANK_INT_TYPE: Vertical Blank Interrupt Type 0 = EDGE 1 = LEVEL
1	0x0	FRAME_END_INT_TYPE: Frame End Interrupt Type 0 = EDGE 1 = LEVEL

24.6.23 DC_CMD_INT_POLARITY_0

For edge interrupts, these bits specify whether a pending interrupt is generated on the falling edge or on the rising edge of the corresponding input signal/event. For level interrupts, these bits specify whether a pending interrupt is generated on the low level or on the high level of the corresponding input signal/event.

Interrupt Polarity

Offset: 0x3b | Byte Offset: 0xec | Read/Write: R/W | Reset: 0x00400000 (0bxx00000001xxxxx00000x0000x00000x)

Bit	Reset	Description
29	0x0	DSC_TO_UF_INT_POLARITY: DSC TO Underflow Interrupt Polarity. 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
28	0x0	DSC_BBUF_UF_INT_POLARITY: DSC BBUF Underflow Interrupt Polarity. 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
27	0x0	DSC_RBUF_UF_INT_POLARITY: DSC RBUF Underflow Interrupt Polarity. 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
26	0x0	DSC_OBUF_UF_INT_POLARITY: DSC OBUF Underflow Interrupt Polarity. 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
25	0x0	WIN_T_UF_INT_POLARITY: Window T Underflow Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
24	0x0	WIN_D_UF_INT_POLARITY: Window D Underflow Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
23	0x0	HC_UF_INT_POLARITY: Cursor Underflow Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
22	HIGH	CMU_LUT_CONFLICT_INT_POLARITY: CMU LUT read/write conflict. Must be HIGH. 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
16	0x0	WIN_C_OF_INT_POLARITY: Window C Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
15	0x0	WIN_B_OF_INT_POLARITY: Window B Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
14	0x0	WIN_A_OF_INT_POLARITY: Window A Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
13	0x0	SSF_INT_POLARITY: Sub-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
12	0x0	MSF_INT_POLARITY: Main-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
10	0x0	WIN_C_UF_INT_POLARITY: Window C Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
9	0x0	WIN_B_UF_INT_POLARITY: Window B Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
8	0x0	WIN_A_UF_INT_POLARITY: Window A Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
7	0x0	SPI_BUSY_INT_POLARITY: SPI Busy. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
5	0x0	V_PULSE2_INT_POLARITY: V Pulse 2 Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
4	0x0	V_PULSE3_INT_POLARITY: V Pulse 3. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
3	0x0	H_BLANK_INT_POLARITY: H Blank. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
2	0x0	V_BLANK_INT_POLARITY: V Blank. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
1	0x0	FRAME_END_INT_POLARITY: Frame End. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

24.6.24 DC_CMD_SIGNAL_RAISE1_0

When written, the next occurrence of the signal in SELECT will cause a raise with a number of VECTOR to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE1_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE1_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE1_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE1_SELECT=NONE or SIGNAL_RAISE1_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE1_SELECT: which signal to raise on. 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE1_VECTOR: bit number to raise

24.6.25 DC_CMD_SIGNAL_RAISE2_0

When written, the next occurrence of the signal in SELECT will cause a raise with a number of VECTOR to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE2_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE2_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE2_TYPE: 0= One-shot, single raise returned 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE2_SELECT=NONE or SIGNAL_RAISE2_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE2_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE2_VECTOR: bit number to raise

24.6.26 DC_CMD_SIGNAL_RAISE3_0

When written, the next occurrence of the signal in SELECT will cause a raise with a number of VECTOR to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE3_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE3_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE3_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE3_SELECT=NONE or SIGNAL_RAISE3_TYPE=ONESHOT 0 = ONESHOT 1 = CONT

Bit	Reset	Description
10:8	X	SIGNAL_RAISE3_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE3_VECTOR: bit number to raise

24.6.27 DC_CMD_STATE_ACCESS_0

Double/triple buffers read and write access control

Offset: 0x40 | Byte Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0)

Bit	Reset	Description
2	0x0	WRITE_MUX: Write access control 0= write assembly state 1= write active state When set to ACTIVE, register writes also propagate to assembly set for double buffered registers, to both assembly and arm set for triple buffered registers. 0 = ASSEMBLY 1 = ACTIVE
0	0x0	READ_MUX: Read access control 0= read assembly state 1= read active state ARM state register read is not controlled by this mux, but by reading the registers with "_NS" suffix 0 = ASSEMBLY 1 = ACTIVE

24.6.28 DC_CMD_STATE_CONTROL_0

State Control for Activating/Arming New Register State

Restrictions: ACT_REQ cannot be programmed at the same time the corresponding "UPDATE" is programmed.

If so desired, it should be split into two consecutive writes to this register.

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx0xxxxxxx0xx000000xx00000)

Bit	Reset	Description
24	0x0	NC_HOST_TRIG_ENABLE: Host trigger enable. Effective only in Non-continuous mode. The exception is that when TVO is enabled, this trigger is ignored so as not to corrupt TV output. Note that when this field is enabled, GENERAL_ACT_REQ must be enabled at the same time. 0= disable: no frame is triggered 0 = DISABLE 1 = ENABLE
15	0x0	CURSOR_UPDATE: Trigger for the arming state (from assembly to armed state) for a subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	WIN_D_UPDATE: Trigger for the arming state (from assembly to armed state) for the win D subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
11	0x0	WIN_C_UPDATE: Trigger for arming state (from assembly to armed state) for the win C subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
10	0x0	WIN_B_UPDATE: Trigger for arming state (from assembly to armed state) for the win B subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
9	0x0	WIN_A_UPDATE: Trigger for arming state (from assembly to armed state) for the win A subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
8	0x0	GENERAL_UPDATE: Trigger for arming state (from assembly to armed state) for a subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
7	0x0	CURSOR_ACT_REQ: Non-window specific 0= no request pending/request completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
4	0x0	WIN_D_ACT_REQ: Window D activation request 0= no request pending/request completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
3	0x0	WIN_C_ACT_REQ: Window C activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
2	0x0	WIN_B_ACT_REQ: Window B activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
1	0x0	WIN_A_ACT_REQ: Window A activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
0	0x0	GENERAL_ACT_REQ: Non-window-specific 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE

24.6.29 DC_CMD_DISPLAY_WINDOW_HEADER_0

Display Window Header for programming display windows and their corresponding buffer start addresses.

Class: Display Window Programming Header

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000xxxx)

Bit	Reset	Description
7	0x0	WINDOW_D_SELECT: Window D Select 0= disable window D programming 1= enable window D programming 0 = DISABLE 1 = ENABLE
6	0x0	WINDOW_C_SELECT: Window C Select. 0= disable window C programming. 1= enable window C programming 0 = DISABLE 1 = ENABLE
5	0x0	WINDOW_B_SELECT: Window B Select. 0= disable window B programming. 1= enable window B programming 0 = DISABLE 1 = ENABLE
4	0x0	WINDOW_A_SELECT: Window A Select. 0= disable window A programming. 1= enable window A programming 0 = DISABLE 1 = ENABLE

24.6.30 DC_CMD_REG_ACT_CONTROL_0

Register activation options

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx0xx00x0x0x0)

Bit	Reset	Description
10	0x0	WIN_D_ACT_CNTR_SEL: Select which counter to use for window D activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
7	0x0	CURSOR_ACT_CNTR_SEL: Select which counter to use for window C activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
6	0x0	WIN_C_ACT_CNTR_SEL: Select which counter to use for window C activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
4	0x0	WIN_B_ACT_CNTR_SEL: Select which counter to use for window B activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
2	0x0	WIN_A_ACT_CNTR_SEL: Select which counter to use for window A activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER
0	0x0	GENERAL_ACT_CNTR_SEL: Select which counter to use for general activation 0= vertical counter 1= horizontal counter 0 = VCOUNTER 1 = HCOUNTER

24.6.31 DC_CMD_WIN_T_STATE_CONTROL_0

WIN_T State control register for activating/arming new register state

Restrictions:

- ACT_REQ cannot be programmed at the same time the corresponding "UPDATE" is programmed. If so desired, it should be split into two consecutive writes to this register.
- When SECURE_CONTROL.SECURE_ENABLE=1, this register is only accessible when HOST1X secure bit is set.
- If SECURE_CONTROL.SECURE_ENABLE is zero, this register is accessible regardless of HOST1X secure bit level.

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_T_UPDATE: Trigger for the arming state (from assembly to armed state) for the win T subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
0	0x0	WIN_T_ACT_REQ: 0= no request pending/request completed 0 = DISABLE 1 = ENABLE

24.6.32 DC_CMD_SECURE_CONTROL_0

Secure control register for enabling secure mode and controlling host1x register loads and reads

Restrictions:

- This register is only accessible when the HOST1X secure bit is set. If the HOST1X secure bit is zero writes to this register are ignored and reads return 0.

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000x00xxxxx000)

Bit	Reset	Description
14	DISABLE	SECURE_SOR1_PROTECT: If ENABLE, blanks the display output to SOR1 when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
13	DISABLE	SECURE_SOR_PROTECT: if ENABLE, blanks the display output to SOR when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
12	DISABLE	SECURE_DSIB_PROTECT: if ENABLE, blanks the display output to DSIB when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
11	DISABLE	SECURE_DSIA_PROTECT: If ENABLE, blanks the display output to DSIA when SECURE_ENABLE is true. 0 = DISABLE 1 = ENABLE
9	ASSEMBLY	SECURE_READ_MUX: Controls which of the double-buffered window registers are read in TrustZone Secure mode (host1x secure bit set). Similar to STATE_ACCESS.READ_MUX 0 = ASSEMBLY 1 = ACTIVE
8	ASSEMBLY	SECURE_WRITE_MUX: Controls which of the double-buffered window registers are written in TrustZone Secure mode (host1x secure bit set). Similar to STATE_ACCESS.WRITE_MUX 0 = ASSEMBLY 1 = ACTIVE
2	DISABLE	SECURE_CRC_PROTECT: If ENABLE, and SECURE_ENABLE=ENABLE, CRC computation is disabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	DISABLE	SECURE_CMU_PROTECT: If ENABLE, and SECURE_ENABLE=ENABLE, CMU registers are only accessible in TrustZone Secure mode. 0 = DISABLE 1 = ENABLE
0	DISABLE	SECURE_ENABLE: Double-buffered: Active state changed at next frame boundary, or immediately in STOP mode. Read returns active state. Changes window T to and from Secure mode 0 = DISABLE 1 = ENABLE

24.6.33 DC_CMD_WIN_D_INCR_SYNCPT_0

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	WIN_D_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = LINE_FLIPPED 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	WIN_D_INDX: syncpt index value

24.6.34 DC_CMD_WIN_D_INCR_SYNCPT_CNTRL_0

Offset: 0x4d | Byte Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	WIN_D_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_D_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

24.6.35 DC_CMD_WIN_D_INCR_SYNCPT_ERROR_0

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_D_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

24.7 Display COM Registers

24.7.1 DC_COM_CRC_CONTROL_0

CRC Control

CRC is provided for at speed testing and diagnostic. When CRC is enabled, the CRC logic waits for the next VSync pulse or the one after that (depending on CRC_WAIT) and then it captures one frame of data at the end of display pipeline and computes the CRC value.

After one frame of data is captured, the CRC logic will stop capturing data.

When CRC_INPUT_DATA = FULL_FRAME, DISPLAY_COMMAND.DISPLAY_CTRL_MODE should be programmed to C_DISPLAY so that CRC works properly.

When CRC_INPUT_DATA = ACTIVE_DATA, it can work on both NC_DISPLAY and C_DISPLAY modes, and can work for multiple frames if CRC is checked, disabled, and re-enabled after the end of frame v-active area and before next VSYNC.

CRC logic takes 8 bits of control signals and 24-bit RGB pixel after dither and after display color (R and B) swap option.

Input [31:0] into CRC depends on CRC_INPUT_DATA. If programmed as FULL_FRAME, input data is {LVP1, LPV0, LHP2, LHP1, LHP0, VSYNC, HSYNC, ACTIVE, R[7:0], G[7:0], B[7:0]} and CRC runs over the entire frame (including blank). If programmed as ACTIVE_DATA, input data is {R[7:0], G[7:0], B[7:0]} and CRC runs only during active display area.

Offset: 0x300 | Byte Offset: 0xc00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	CRC_ALWAYS: CRC always: calculates CRC for every following frame. Must use with CRC_INPUT_DATA = ACTIVE_DATA. If enabled, CRC_WAIT field is ignored. 0 = DISABLE 1 = ENABLE
2	0x0	CRC_INPUT_DATA: CRC input data 0= Full frame (RGB data and control) 1= Active display (Only RGB data) 0 = FULL_FRAME 1 = ACTIVE_DATA
1	0x0	CRC_WAIT: CRC Wait 0= 1 Vsync 1= 2 Vsyncs
0	0x0	CRC_ENABLE: CRC Enable 0 = DISABLE 1 = ENABLE

24.7.2 DC_COM_CRC_CHECKSUM_0

CRC Checksum

This register can be read by host after CRC logic stops capturing data.

Offset: 0x301 | Byte Offset: 0xc04 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM: CRC Checksum

24.7.3 DC_COM_PIN_MISC_CONTROL_0

Pin Miscellaneous Control

Pin Miscellaneous Control

Offset: 0x31b | Byte Offset: 0xc6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx)

Bit	Reset	Description
2	0x0	DISP_CLOCK_OUTPUT: Display Clock (DCLK) Enable. 0= disable. 1= enable display clock to be output on LCD_DE pin (LCD_DE output select must be appropriately programmed for this to be effective) 0 = DISABLE 1 = ENABLE

24.7.4 DC_COM_PM0_CONTROL_0

PM0 Signal Control

Class: Pulse Width Modulation

PM0 signal is programmable pulse width modulation signal that can be output on several pins. The control register should be initialized once before PM0 is enabled.

Note: *The period is expressed as multiples of 4 times the divider value.*
The actual period - in clock cycles - is given by:
*period = (1 + PM0_CLOCK_DIVIDER) * ((PM0_PERIOD + 1) * 4)*

Offset: 0x31c | Byte Offset: 0xc70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:18	X	PM0_PERIOD: PM0 Period (4, 8, ... , 256 clock cycles)
17:4	X	PM0_CLOCK_DIVIDER: PM0 Clock Divider (1 to 16384)
1:0	X	PM0_CLOCK_SELECT: PM0 Clock Select. 0= output of shift clock divider 1= pixel clock 2= line clock 3= frame clock Notes: 1) Pixel clock, line clock, and frame clock are running only when the PW0 signal is enabled. 2) In non-continuous mode, shift clock and pixel clock run continuously, but line clock and frame clock only run while a frame is being sent.

24.7.5 DC_COM_PM0_DUTY_CYCLE_0

PM0 Duty Cycle

A counter repeatedly counts up from 0 to ((PM0_PERIOD << 2) + 3) pre-scaled cycles.

The period always starts with the output value == 1. After DUTY_CYCLE number of pre-scaled cycles, the output value is cleared for the remainder of the period. In order to output constant 0, simply set the DUTY_CYCLE to 0. To output a constant 1, set DUTY_CYCLE to be any value greater than ((PM0_PERIOD << 2) + 3).

Offset: 0x31d | Byte Offset: 0xc74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8:0	X	PM0_DUTY_CYCLE: PM0 Duty Cycle (or D) From 1/P to D/P pulse high time where P is the period.

24.7.6 DC_COM_SCRATCH_REGISTER_A_0

Scratch Register A

Class: Software Scratch Registers

Offset: 0x325 | Byte Offset: 0xc94 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_A: Scratch Register A

24.7.7 DC_COM_SCRATCH_REGISTER_B_0

Scratch Register B

Offset: 0x326 | Byte Offset: 0xc98 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_B: Scratch Register B

24.7.8 DC_COM_CRC_CHECKSUM_LATCHED_0

CRC Checksum latched

This register is a latched version of CRC_CHECKSUM. Latching happens at frame end.

Note: *CRC_INPUT_DATA needs to be set to ACTIVE_DATA if this register is used. In full frame mode, CRC is frozen two cycles after frame end due to pipelining, so only in active area mode, CRC is consistent and independent of display control mode, and can be checked continuously frame by frame.*

Offset: 0x329 | Byte Offset: 0xca4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM_LATCHED: CRC Checksum latched

24.7.9 DC_COM_CMU_CSC_KRR_0

CMU Color Space Conversion Matrix

- R' = sat(KRR * R + KGR * G + KBR * B)
- G' = sat(KRG * R + KGG * G + KBG * B)
- B' = sat(KRB * R + KGB * G + KBB * B)

Coefficients are signed 1.8 fixed point, for a range of approximately [-2.0, +1.996]

Offset: 0x32a | Byte Offset: 0xca8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRR

24.7.10 DC_COM_CMU_CSC_KGR_0

Offset: 0x32b | Byte Offset: 0xcac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGR

24.7.11 DC_COM_CMU_CSC_KBR_0

Offset: 0x32c | Byte Offset: 0xcb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBR

24.7.12 DC_COM_CMU_CSC_KRG_0

Offset: 0x32d | Byte Offset: 0xcb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRG

24.7.13 DC_COM_CMU_CSC_KGG_0

Offset: 0x32e | Byte Offset: 0xcb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGG

24.7.14 DC_COM_CMU_CSC_KBG_0

Offset: 0x32f | Byte Offset: 0xcbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBG

24.7.15 DC_COM_CMU_CSC_KRB_0

Offset: 0x330 | Byte Offset: 0xcc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRB

24.7.16 DC_COM_CMU_CSC_KGB_0

Offset: 0x331 | Byte Offset: 0xcc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGB

24.7.17 DC_COM_CMU_CSC_KBB_0

Offset: 0x332 | Byte Offset: 0xcc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBB

24.7.18 DC_COM_CMU_LUT_MASK_0

CMU LUT Mask

LUT channel data write enables

Offset: 0x333 | Byte Offset: 0xccc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx00xxxxxx00)

Bit	Reset	Description
9:8	0x0	LUT2_WR_MASK: Color channel write mask 0 = enable all channels 1-3 = enable red/green/blue only 0 = ALL 1 = RED 2 = GREEN 3 = BLUE
1:0	0x0	LUT1_WR_MASK: Color channel write mask 0 = enable all channels 1-3 = enable red/green/blue only 0 = ALL 1 = RED 2 = GREEN 3 = BLUE

24.7.19 DC_COM_CMU_LUT1_0

Entries are written indirectly.

Ensure CMU_ENABLE=DISABLE or display idle

For C = 0 .. 255:

C' = ungamma_function(C/255) * ((1 << NV_DISPLAY_CMU_DP_WIDTH) - 1)

Write CMU_LUT1.LUT1_ADDR = C, LUT1_DATA = C'

Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE

Use GENERAL_ACT_REQ to activate CMU_ENABLE

CMU LUT1

Offset: 0x336 | Byte Offset: 0xcd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
27:16	X	LUT1_DATA: Readable only when LUT1_READ_EN=1
7:0	0x0	LUT1_ADDR

24.7.20 DC_COM_CMU_LUT2_0

Write sequence:

Ensure CMU_ENABLE=DISABLE or display idle

For C = 0 .. NV_DISPLAY_CMU_LUT2_DEPTH-1:

C' = gamma_function(inverse_zone_select(C)/((1 << NV_DISPLAY_CMU_DP_WIDTH) - 1)) * 255

Write CMU_LUT2.LUT2_ADDR = C, LUT2_DATA = C'

Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE

Use GENERAL_ACT_REQ

CMU LUT2

Offset: 0x337 | Byte Offset: 0xcdc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
23:16	X	LUT2_DATA: Readable only when LUT2_READ_EN=1
9:0	0x0	LUT2_ADDR

24.7.21 DC_COM_DSC_TOP_CTL_0

Offset: 0x33e | Byte Offset: 0xcf8 | Read/Write: R/W | Reset: 0x000ffff0 (0bxxxxxxxxxxxxx11111111111111110000)

Bit	Reset	Description
19:4	0xffff	DSC_TIMEOUT_COUNTER
3	0x0	DSC_AUTO_RESET
2	0x0	DSC_SLCG_OVERRIDE
1	0x0	DSC_ENABLE
0	0x0	DSC_SOFT_RESET

24.7.22 DC_COM_DSC_DELAY_0

Offset: 0x33f | Byte Offset: 0xcfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DSC_OUTPUT_DELAY

24.7.23 DC_COM_DSC_COMMON_CTL_0

Offset: 0x340 | Byte Offset: 0xd00 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxxxx00000000000)

Bit	Reset	Description
31:16	0x0	DSC_CHUNK_SIZE
10	0x0	DSC_BLOCK_PRED_ENABLE
9:0	0x0	DSC_BITS_PER_PIXEL

24.7.24 DC_COM_DSC_SLICE_INFO_0

Offset: 0x341 | Byte Offset: 0xd04 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_SLICE_HEIGHT: Slice height in pixels. When the whole picture is one slice, the slice height should equal the picture height.
15:0	0x0	DSC_SLICE_WIDTH: Slice width in pixels. When the whole picture is one slice, the slice width should equal the picture width.

24.7.25 DC_COM_DSC_RC_DELAY_INFO_0

Offset: 0x342 | Byte Offset: 0xd08 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxxxxx0000000000)

Bit	Reset	Description
31:16	0x0	DSC_INITIAL_DEC_DELAY: Number of pixels to delay the VLD on the decoder, not including SSM
9:0	0x0	DSC_INITIAL_XMIT_DELAY: Number of pixels to delay the initial transmission

24.7.26 DC_COM_DSC_RC_SCALE_INFO_0

Offset: 0x343 | Byte Offset: 0xd0c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000000000000000000000)

Bit	Reset	Description
21:6	0x0	DSC_SCALE_DECR_INTERVAL: Decrement scale factor every scale_decr_interval group
5:0	0x0	DSC_INITIAL_SCALE_VALUE: Initial value for scale factor

24.7.27 DC_COM_DSC_RC_SCALE_INFO_2_0

Offset: 0x344 | Byte Offset: 0xd10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DSC_SCALE_INCR_INTERVAL: Increment scale factor every scale_incr_interval group

24.7.28 DC_COM_DSC_RC_BPGOFF_INFO_0

Offset: 0x345 | Byte Offset: 0xd14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_SLICE_BPG_OFFSET: BPG offset used to enforce slice bit constraint
15:0	0x0	DSC_NFL_BPG_OFFSET: Non-first line BPG offset to use

24.7.29 DC_COM_DSC_RC_OFFSET_INFO_0

Offset: 0x346 | Byte Offset: 0xd18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_FINAL_OFFSET: Final RC linear transformation offset value
15:0	0x0	DSC_INITIAL_OFFSET: Value to use for RC model offset at slice start

24.7.30 DC_COM_DSC_RC_FLATNESS_INFO_0

Offset: 0x347 | Byte Offset: 0xd1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx000000000000000)

Bit	Reset	Description
14:10	**0x0**	**DSC_FIRST_LINE_BPG_OFFS: BPG offset to use for first line of the slice**
9:5	0x0	DSC_FLATNESS_MAX_QP: Maximum QP where flatness information is sent
4:0	0x0	DSC_FLATNESS_MIN_QP: Minimum QP where flatness information is sent

24.7.31 DC_COM_DSC_RC_PARAM_SET_0

Offset: 0x348 | Byte Offset: 0xd20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
21:18	0x0	DSC_RC_TGT_OFFSET_LO: Offset to BPG used by RC to determine QP adjustment

Bit	Reset	Description
17:14	0x0	DSC_RC_TGT_OFFSET_HI: Offset to BPG used by RC to determine QP adjustment
13:9	0x0	DSC_RC_QUANT_INCR_LIMIT1: Slow down incrementing once the range reaches this value
8:4	0x0	DSC_RC_QUANT_INCR_LIMIT0: Slow down incrementing once the range reaches this value
3:0	0x0	DSC_RC_EDGE_FACTOR: Factor to determine if an edge is present based on the bits produced

24.7.32 DC_COM_DSC_RC_BUF_THRESH0_0

Offset: 0x349 | Byte Offset: 0xd24 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DSC_RC_BUF_THRESH1:Threshold1 defining the buffer ranges
23:16	0x0	DSC_RC_BUF_THRESH0:Threshold0 defining the buffer ranges
15:0	0x0	DSC_RC_MODEL_SIZE: Total size of RC model

24.7.33 DC_COM_DSC_RC_BUF_THRESH1_0

Offset: 0x34a | Byte Offset: 0xd28 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DSC_RC_BUF_THRESH5:Threshold5 defining the buffer ranges
23:16	0x0	DSC_RC_BUF_THRESH4:Threshold4 defining the buffer ranges
15:8	0x0	DSC_RC_BUF_THRESH3:Threshold3 defining the buffer ranges
7:0	0x0	DSC_RC_BUF_THRESH2:Threshold2 defining the buffer ranges

24.7.34 DC_COM_DSC_RC_BUF_THRESH2_0

Offset: 0x34b | Byte Offset: 0xd2c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DSC_RC_BUF_THRESH9:Threshold9 defining the buffer ranges
23:16	0x0	DSC_RC_BUF_THRESH8:Threshold8 defining the buffer ranges
15:8	0x0	DSC_RC_BUF_THRESH7:/Threshold7 defining the buffer ranges
7:0	0x0	DSC_RC_BUF_THRESH6:Threshold6 defining the buffer ranges

24.7.35 DC_COM_DSC_RC_BUF_THRESH3_0

Offset: 0x34c | Byte Offset: 0xd30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DSC_RC_BUF_THRESH13:Threshold13 defining the buffer ranges
23:16	0x0	DSC_RC_BUF_THRESH12:Threshold12 defining the buffer ranges
15:8	0x0	DSC_RC_BUF_THRESH11:Threshold11 defining the buffer ranges
7:0	0x0	DSC_RC_BUF_THRESH10:Threshold10 defining the buffer ranges

24.7.36 DC_COM_DSC_RC_RANGE_CFG0_0

Offset: 0x34d | Byte Offset: 0xd34 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_RC_RANGE_PARAM1:Parameters for RC range1 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset
15:0	0x0	DSC_RC_RANGE_PARAM0:Parameters for RC range0 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset

24.7.37 DC_COM_DSC_RC_RANGE_CFG1_0

Offset: 0x34e | Byte Offset: 0xd38 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_RC_RANGE_PARAM3:Parameters for RC range3 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset
15:0	0x0	DSC_RC_RANGE_PARAM2:Parameters for RC range2 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset

24.7.38 DC_COM_DSC_RC_RANGE_CFG2_0

Offset: 0x34f | Byte Offset: 0xd3c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_RC_RANGE_PARAM5: Parameters for RC range5 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset
15:0	0x0	DSC_RC_RANGE_PARAM4: Parameters for RC range4 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset

24.7.39 DC_COM_DSC_RC_RANGE_CFG3_0

Offset: 0x350 | Byte Offset: 0xd40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_RC_RANGE_PARAM7: Parameters for RC range7 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset
15:0	0x0	DSC_RC_RANGE_PARAM6: Parameters for RC range6 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset

24.7.40 DC_COM_DSC_RC_RANGE_CFG4_0

Offset: 0x351 | Byte Offset: 0xd44 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_RC_RANGE_PARAM9: Parameters for RC range9 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset
15:0	0x0	DSC_RC_RANGE_PARAM8: Parameters for RC range8 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset

24.7.41 DC_COM_DSC_RC_RANGE_CFG5_0

Offset: 0x352 | Byte Offset: 0xd48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_RC_RANGE_PARAM11: Parameters for RC range11 4:0 : range_min_qp 9:5 : range_max_qp 15:10 range_bpg_offset
15:0	0x0	DSC_RC_RANGE_PARAM10: Parameters for RC range10 4:0 : range_min_qp 9:5 : range_max_qp 15:10 range_bpg_offset

24.7.42 DC_COM_DSC_RC_RANGE_CFG6_0

Offset: 0x353 | Byte Offset: 0xd4c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	DSC_RC_RANGE_PARAM13: Parameters for RC range13 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset
15:0	0x0	DSC_RC_RANGE_PARAM12: Parameters for RC range12 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset

24.7.43 DC_COM_DSC_RC_RANGE_CFG7_0

Offset: 0x354 | Byte Offset: 0xd50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DSC_RC_RANGE_PARAM14: Parameters for RC range14 4:0: range_min_qp 9:5: range_max_qp 15:10 range_bpg_offset

24.7.44 DC_COM_DSC_UNIT_SET_0

Offset: 0x355 | Byte Offset: 0xd54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	DSC_LINEBUF_DEPTH: 0: 9 bits 1: 8 bits
1:0	0x0	DSC_SLICE_NUM_MINUS1_IN_LINE: The slice number minus 1 in the line, maximum is 4

24.7.45 DC_COM_DSC_CRC_CONTROL_0

Offset: 0x356 | Byte Offset: 0xd58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DSC_CRC_ALWAYS: CRC ALWAYS 0= disable 1= enable 0 = DISABLE 1 = ENABLE
0	0x0	DSC_CRC_ENABLE: CRC Enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE

24.7.46 DC_COM_DSC_CRC_CHECKSUM_0

CRC Checksum latched

Offset: 0x357 | Byte Offset: 0xd5c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSC_CRC_CHECKSUM: CRC Checksum latched

24.7.47 DC_COM_DSC_STATUS_0

Offset: 0x358 | Byte Offset: 0xd60 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
17:16	X	DSC_STATUS_SLICEID
15:0	X	DSC_STATUS_HINDEX

24.7.48 DC_COM_DSC_STATUS_2_0

Offset: 0x359 | Byte Offset: 0xd64 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	DSC_STATUS_BUSY

Bit	Reset	Description
15:0	X	DSC_STATUS_VINDEX

24.8 Display DISP Registers

These registers control DISP display only; not including the DISP display window parameters.

24.8.1 DC_DISP_DISP_SIGNAL_OPTIONS0_0

Display Signal Options 0

Offset: 0x400 | Byte Offset: 0x1000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0x0xxx000x0xxx0x0x0xxxxxxxx)

Bit	Reset	Description
26	0x0	M1_ENABLE: M1 Enable 0 = DISABLE 1 = ENABLE
24	0x0	M0_ENABLE: M0 Enable 0 = DISABLE 1 = ENABLE
20	0x0	V_PULSE3_ENABLE: V Pulse 3 Enable 0 = DISABLE 1 = ENABLE
19	0x0	V_PULSE2_ENABLE: V Pulse 2 Enable 0 = DISABLE 1 = ENABLE
18	0x0	V_PULSE1_ENABLE: V Pulse 1 Enable 0 = DISABLE 1 = ENABLE
16	0x0	V_PULSE0_ENABLE: V Pulse 0 Enable 0 = DISABLE 1 = ENABLE
12	0x0	H_PULSE2_ENABLE: H Pulse 2 Enable 0 = DISABLE 1 = ENABLE
10	0x0	H_PULSE1_ENABLE: H Pulse 1 Enable 0 = DISABLE 1 = ENABLE
8	0x0	H_PULSE0_ENABLE: H Pulse 0 Enable 0 = DISABLE 1 = ENABLE

24.8.2 DC_DISP_DISP_WIN_OPTIONS_0

Display Window Options

Offset: 0x402 | Byte Offset: 0x1008 | Read/Write: R/W | Reset: 0x00000000 (0bxx0x000xxxxxxxx0xxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	0x0	DSI_ENABLE: MIPI Display Serial Interface Enable. The DSI unit must also be separately enabled in its own register space in order to use DSI functionality. 0 = DISABLE 1 = ENABLE
27	DP	SOR1_TIMING_CYA: HDMI expects a delayed vsync and preamble compared to DP 0 = DP 1 = HDMI

Bit	Reset	Description
26	0x0	SOR1_ENABLE: SOR1 interface - DP/HDMI 0 = disable 1 = enable The SOR1 unit must also be separately enabled in its own register space in order to use SOR1 functionality. 0 = DISABLE 1 = ENABLE
25	0x0	SOR_ENABLE: SOR interface - eDP The SOR unit must also be separately enabled in its own register space in order to use SOR functionality. 0 = DISABLE 1 = ENABLE
16	0x0	CURSOR_ENABLE: Cursor Enable 0 = DISABLE 1 = ENABLE

24.8.3 DC_DISP_DISP_TIMING_OPTIONS_0

Display Timing Options

Class: Display Standard Timings

Programming of display timing registers must meet these restrictions:

- Constraint 1: H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH > 20.
- Constraint 2: V_REF_TO_SYNC + V_SYNC_WIDTH + V_BACK_PORCH > 1.
- Constraint 3: V_FRONT_PORCH + V_SYNC_WIDTH + V_BACK_PORCH > 1 (vertical blank).
- Constraint 4: V_SYNC_WIDTH >= 1

  H_SYNC_WIDTH >= 1
- Constraint 5: V_REF_TO_SYNC >= 1

  H_REF_TO_SYNC >= 0
- Constraint 6: V_FRONT_PORCH >= (V_REF_TO_SYNC + 1)

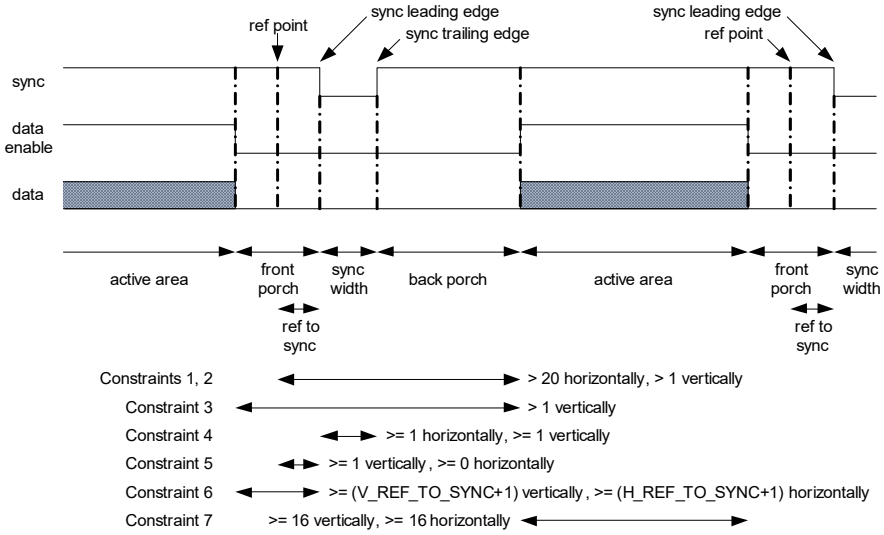
  H_FRONT_PORCH >= (H_REF_TO_SYNC + 1)
- Constraint 7: H_DISP_ACTIVE >= 16

  V_DISP_ACTIVE >= 16

Timing Diagram

- This diagram applies to both vertical and horizontal timing
- The back porch is the only parameter that can be negative

Figure 99: Display Timing Options Timing Diagram



This register specifies display timing options for HSYNC and VSYNC.

Offset: 0x405 | Byte Offset: 0x1014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	VSYNC_H_POSITION: VSYNC Horizontal Position. This parameter specifies the position where VSYNC can toggle with respect to H reference point.

24.8.4 DC_DISP_REF_TO_SYNC_0

H/V Reference to Sync

This register specifies the start position of HSYNC and VSYNC with respect to H and V reference point (line and frame start) correspondingly. The H and V reference points correspond to the time when H and V display timing counter is re-initialized to zero correspondingly. Check DC_DISP_DISP_TIMING_OPTIONS_0, the Display Timing Options register, for programming restrictions for REF_TO_SYNC.

The H reference point also determines the point where V display timing counter is incremented so this points the horizontal relationship between HSYNC and VSYNC.

Note: *VSYNC's rising/falling edge is fixed at H reference point zero.*

Offset: 0x406 | Byte Offset: 0x1018 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_REF_TO_SYNC: V reference to VSYNC (minimum 1 line clock)
12:0	0x0	H_REF_TO_SYNC: H reference to HSYNC (minimum 0 pixel clock)

24.8.5 DC_DISP_SYNC_WIDTH_0

H/V SYNC Pulse Width

This register specifies the width of HSYNC and VSYNC pulses. Check DC_DISP_DISP_TIMING_OPTIONS_0, the Display Timing Options register, for programming restrictions for SYNC_WIDTH.

Offset: 0x407 | Byte Offset: 0x101c | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_SYNC_WIDTH: VSYNC pulse width (minimum 1 line clock)
12:0	0x0	H_SYNC_WIDTH: HSYNC pulse width (minimum 1 pixel clock)

24.8.6 DC_DISP_BACK_PORCH_0

H/V Back Porch

This register specifies the distance between H/V SYNC trailing edge to beginning of display active area. This is a 2's complement value, and a negative value indicates that H/V SYNC overlaps with the corresponding display active area. Check DC_DISP_DISP_TIMING_OPTIONS_0, the Display Timing Options register, for programming restrictions for BACK_PORCH.

Offset: 0x408 │ Byte Offset: 0x1020 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_BACK_PORCH: V back porch
12:0	0x0	H_BACK_PORCH: H back porch

24.8.7 DC_DISP_DISP_ACTIVE_0

H/V Display Active Width

This register specifies the width of the H/V display active area. Check DC_DISP_DISP_TIMING_OPTIONS_0, the Display Timing Options register, for programming restrictions for DISP_ACTIVE.

Offset: 0x409 │ Byte Offset: 0x1024 │ Read/Write: R/W │ Reset: 0x00030003 (0bxxx0000000000011xxx0000000000011)

Bit	Reset	Description
28:16	0x3	V_DISP_ACTIVE: V display active width (minimum 16 lines)
12:0	0x3	H_DISP_ACTIVE: H display active width (minimum 16 pixels)

24.8.8 DC_DISP_FRONT_PORCH_0

H/V Front Porch

This register specifies the distance between end of H/V display active area to the leading edge of the corresponding H/V SYNC.

Design Note: H/V active end plus the H/V front porch value minus the H/V reference to H/VSYNC determines the H/V total (final H/V count value for the H/V display counter). Check DC_DISP_DISP_TIMING_OPTIONS_0, the Display Timing Options register, for programming restrictions for FRONT_PORCH.

Offset: 0x40a │ Byte Offset: 0x1028 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	V_FRONT_PORCH: VSYNC front porch (minimum -=PS_=-V_REF_TO_SYNC + 1)
12:0	0x0	H_FRONT_PORCH: HSYNC front porch (minimum -=PS_=-H_REF_TO_SYNC + 1)

24.8.9 DC_DISP_H_PULSE0_CONTROL_0

H Pulse 0 Control

Class: Display Extended Timings

Horizontal pulse 0 is programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop

as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode, this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 0.

Offset: 0x40b │ Byte Offset: 0x102c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE0_LAST: H Pulse 0 Last point. 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE0_V_QUAL: H Pulse 0 Vertical Qualifier. 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE0_POLARITY: H Pulse 0 Polarity. Polarity adjustment is done before the vertical qualifier is applied. 0 = HIGH 1 = LOW
3	X	H_PULSE0_MODE: H Pulse 0 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

24.8.10 DC_DISP_H_PULSE0_POSITION_A_0

H Pulse 0 Position A

Offset: 0x40c │ Byte Offset: 0x1030 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_A: H Pulse 0 End A (minimum -=PS_=-H_PULSE0_START_A+1)
12:0	X	H_PULSE0_START_A: H Pulse 0 Start A (minimum 0)

24.8.11 DC_DISP_H_PULSE0_POSITION_B_0

H Pulse 0 Position B

Offset: 0x40d | Byte Offset: 0x1034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_B: H Pulse 0 End B (minimum -=PS_=-H_PULSE0_START_B+1)
12:0	X	H_PULSE0_START_B: H Pulse 0 Start B (minimum -=PS_=-H_PULSE0_END_A+1)

24.8.12 DC_DISP_H_PULSE0_POSITION_C_0

H Pulse 0 Position C

Offset: 0x40e | Byte Offset: 0x1038 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_C: H Pulse 0 End C (minimum -=PS_=-H_PULSE0_START_C+1)
12:0	X	H_PULSE0_START_C: H Pulse 0 Start C (minimum -=PS_=-H_PULSE0_END_B+1)

24.8.13 DC_DISP_H_PULSE0_POSITION_D_0

H Pulse 0 Position D

Offset: 0x40f | Byte Offset: 0x103c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_D: H Pulse 0 End D (minimum -=PS_=-H_PULSE0_START_D+1)
12:0	X	H_PULSE0_START_D: H Pulse 0 Start D (minimum -=PS_=-H_PULSE0_END_C+1)

24.8.14 DC_DISP_H_PULSE1_CONTROL_0

H Pulse 1 Control

Horizontal pulse 1 is programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode, this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value), then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 1.

Offset: 0x410 | Byte Offset: 0x1040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE1_LAST: H Pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE1_V_QUAL: H Pulse 1 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE1_POLARITY: H Pulse 1 Polarity. Polarity adjustment is done before the vertical qualifier is applied 0 = HIGH 1 = LOW
3	X	H_PULSE1_MODE: H Pulse 1 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

24.8.15 DC_DISP_H_PULSE1_POSITION_A_0

H Pulse 1 Position A

Offset: 0x411 | Byte Offset: 0x1044 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_A: H Pulse 1 End A (minimum -=PS_=-H_PULSE1_START_A+1)
12:0	X	H_PULSE1_START_A: H Pulse 1 Start A (minimum 0)

24.8.16 DC_DISP_H_PULSE1_POSITION_B_0

H Pulse 1 Position B

Offset: 0x412 | Byte Offset: 0x1048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_B: H Pulse 1 End B (minimum -=PS_=-H_PULSE1_START_B+1)
12:0	X	H_PULSE1_START_B: H Pulse 1 Start B (minimum -=PS_=-H_PULSE1_END_A+1)

24.8.17 DC_DISP_H_PULSE1_POSITION_C_0

H Pulse 1 Position C

Offset: 0x413 | Byte Offset: 0x104c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_C: H Pulse 1 End C (minimum -=PS_=-H_PULSE1_START_C+1)
12:0	X	H_PULSE1_START_C: H Pulse 1 Start C (minimum -=PS_=-H_PULSE1_END_B+1)

24.8.18 DC_DISP_H_PULSE1_POSITION_D_0

H Pulse 1 Position D

Offset: 0x414 │ Byte Offset: 0x1050 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_D: H Pulse 1 End D (minimum -=PS_=-H_PULSE1_START_D+1)
12:0	X	H_PULSE1_START_D: H Pulse 1 Start D (minimum -=PS_=-H_PULSE1_END_C+1)

24.8.19 DC_DISP_H_PULSE2_CONTROL_0

H Pulse 2 Control

Horizontal pulse 2 is a programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value), then the pulse generator will stop at the last valid position. Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 2.

Offset: 0x415 │ Byte Offset: 0x1054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE2_LAST: H Pulse 2 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE2_V_QUAL: H Pulse 2 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1

Bit	Reset	Description
4	X	H_PULSE2_POLARITY: H Pulse 2 Polarity. Polarity adjustment is done before the vertical qualifier is applied 0 = HIGH 1 = LOW
3	X	H_PULSE2_MODE: H Pulse 2 Mode. 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

24.8.20 DC_DISP_H_PULSE2_POSITION_A_0

H Pulse 2 Position A

Offset: 0x416 | Byte Offset: 0x1058 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_A: H Pulse 2 End A (minimum -=PS_=-H_PULSE2_START_A+1)
12:0	X	H_PULSE2_START_A: H Pulse 2 Start A (minimum 0)

24.8.21 DC_DISP_H_PULSE2_POSITION_B_0

H Pulse 2 Position B

Offset: 0x417 | Byte Offset: 0x105c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_B: H Pulse 2 End B (minimum -=PS_=-H_PULSE2_START_B+1)
12:0	X	H_PULSE2_START_B: H Pulse 2 Start B (minimum -=PS_=-H_PULSE2_END_A+1)

24.8.22 DC_DISP_H_PULSE2_POSITION_C_0

H Pulse 2 Position C

Offset: 0x418 | Byte Offset: 0x1060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_C: H Pulse 2 End C (minimum -=PS_=-H_PULSE2_START_C+1)
12:0	X	H_PULSE2_START_C: H Pulse 2 Start C (minimum -=PS_=-H_PULSE2_END_B+1)

24.8.23 DC_DISP_H_PULSE2_POSITION_D_0

H Pulse 2 Position D

Offset: 0x419 | Byte Offset: 0x1064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_D: H Pulse 2 End D (minimum -=PS_=-H_PULSE2_START_D+1)
12:0	X	H_PULSE2_START_D: H Pulse 2 Start D (minimum -=PS_=-H_PULSE2_END_C+1)

24.8.24 DC_DISP_V_PULSE0_CONTROL_0

V Pulse 0 Control

Vertical pulse 0 is a programmable pulse that repeats every frame.

This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 0.

Offset: 0x41a │ Byte Offset: 0x1068 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE0_H_POSITION: V Pulse 0 Horizontal Position. This parameter specifies the position where V Pulse 0 can toggle with respect to H reference point.
11:8	X	V_PULSE0_LAST: V Pulse 0 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C
7:6	X	V_PULSE0_DELAY: V Pulse 0 Delay 0= no delay 1= 1-line delay 2= 2-line delay 3= reserved 0 = NODELAY 1 = DELAY1 2 = DELAY2
4	X	V_PULSE0_POLARITY: V Pulse 0 Polarity 0 = HIGH 1 = LOW

24.8.25 DC_DISP_V_PULSE0_POSITION_A_0

V Pulse 0 Position A

Offset: 0x41b │ Byte Offset: 0x106c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_A: V Pulse 0 End A (minimum -=PS_=-V_PULSE0_START_A+1)
12:0	X	V_PULSE0_START_A: V Pulse 0 Start A (minimum 0)

24.8.26 DC_DISP_V_PULSE0_POSITION_B_0

V Pulse 0 Position B

Offset: 0x41c │ Byte Offset: 0x1070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_B: V Pulse 0 End B (minimum -=PS_=-V_PULSE0_START_B+1)
12:0	X	V_PULSE0_START_B: V Pulse 0 Start B (minimum -=PS_=-V_PULSE0_END_A+1)

24.8.27 DC_DISP_V_PULSE0_POSITION_C_0

V Pulse 0 Position C

Offset: 0x41d │ Byte Offset: 0x1074 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_C: V Pulse 0 End C (minimum -=PS_=-V_PULSE0_START_C+1)
12:0	X	V_PULSE0_START_C: V Pulse 0 Start C (minimum -=PS_=-V_PULSE0_END_B+1)

24.8.28 DC_DISP_V_PULSE1_CONTROL_0

V pulse 1 Control

Vertical pulse 1 is a programmable pulse that repeats every frame. This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 1.

Offset: 0x41e │ Byte Offset: 0x1078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE1_H_POSITION: V Pulse 1 Horizontal Position. This parameter specifies the position where V Pulse 1 can toggle with respect to H reference point.
11:8	X	V_PULSE1_LAST: V pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C
7:6	X	V_PULSE1_DELAY: V pulse 1 Delay 0= no delay 1= 1-line delay 2= 2-line delay 3= reserved 0 = NODELAY 1 = DELAY1 2 = DELAY2
4	X	V_PULSE1_POLARITY: V pulse 1 Polarity 0 = HIGH 1 = LOW

24.8.29 DC_DISP_V_PULSE1_POSITION_A_0

V Pulse 1 Position A

Offset: 0x41f | Byte Offset: 0x107c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_A: V Pulse 1 End A (minimum -=PS_=-V_PULSE1_START_A+1)
12:0	X	V_PULSE1_START_A: V Pulse 1 Start A (minimum 0)

24.8.30 DC_DISP_V_PULSE1_POSITION_B_0

V Pulse 1 Position B

Offset: 0x420 | Byte Offset: 0x1080 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_B: V Pulse 1 End B (minimum -=PS_=-V_PULSE1_START_B+1)
12:0	X	V_PULSE1_START_B: V Pulse 1 Start B (minimum -=PS_=-V_PULSE1_END_A+1)

24.8.31 DC_DISP_V_PULSE1_POSITION_C_0

V Pulse 1 Position C

Offset: 0x421 | Byte Offset: 0x1084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_C: V Pulse 1 End C (minimum -=PS_=-V_PULSE1_START_C+1)
12:0	X	V_PULSE1_START_C: V Pulse 1 Start C (minimum -=PS_=-V_PULSE1_END_B+1)

24.8.32 DC_DISP_V_PULSE2_CONTROL_0

V Pulse 2 Control

Vertical pulse 2 is a programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 2.

Offset: 0x422 | Byte Offset: 0x1088 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE2_H_POSITION: V Pulse 2 Horizontal Position. This parameter specifies the position where V Pulse 2 can toggle with respect to the H reference point.
8	X	V_PULSE2_LAST: V pulse 2 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A

Bit	Reset	Description
4	0x0	V_PULSE2_POLARITY: V pulse 2 Polarity 0 = HIGH 1 = LOW

24.8.33 DC_DISP_V_PULSE2_POSITION_A_0

V Pulse 2 Position A

Offset: 0x423 | Byte Offset: 0x108c | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	X	V_PULSE2_END_A: V Pulse 2 End A (minimum -=PS_=-V_PULSE2_START_A+1)
12:0	X	V_PULSE2_START_A: V Pulse 2 Start A (minimum 0)

24.8.34 DC_DISP_V_PULSE3_CONTROL_0

V Pulse 3 Control

Vertical pulse 3 is a programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 3.

Offset: 0x424 | Byte Offset: 0x1090 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE3_H_POSITION: V Pulse 3 Horizontal Position. This parameter specifies the position where V Pulse 3 can toggle with respect to H reference point.
8	X	V_PULSE3_LAST: V pulse 3 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A
4	0x0	V_PULSE3_POLARITY: V pulse 3 Polarity 0 = HIGH 1 = LOW

24.8.35 DC_DISP_V_PULSE3_POSITION_A_0

V Pulse 3 Position A

Offset: 0x425 | Byte Offset: 0x1094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE3_END_A: V Pulse 3 End A (minimum -=PS_=-V_PULSE3_START_A+1)
12:0	X	V_PULSE3_START_A: V Pulse 3 Start A (minimum 0)

24.8.36 DC_DISP_DISP_CLOCK_CONTROL_0

Display Clock Control

The shift clock divider is used to divide root clock for display controller module to generate internal shift clock for shifting data to the display. Output of this divider is typically used to generate the external shift clock which is sent to the display (SC0 and/or SC1) except for 1-pixel/1-clock parallel display.

The output of this divider is also used to generate Programmable Pulse (PP) signal. For 1-pixel/1-clock parallel display, SC0 and SC1 are generated using the output of pixel clock divider which can be set to 1, 2, or 4 for 1-pixel/1-clock parallel display.

The reason pixel clock divider 2 and 4 are allowed for 1-pixel/1-clock parallel display interface is so that the clock that generates PP can be generated with 2x or 4x higher frequency than pixel clock and therefore can produce higher resolution PP pulse positions. For all cases of parallel display, SC0 and SC1 can be further divided by 1, 2, or 4.

Class: Display Interface Settings

This register controls generation of shift clock to the display and internal pixel clock. Internal display pipeline runs with pixel clock and processes 1 pixel per clock.

Offset: 0x42e │ Byte Offset: 0x10b8 │ Read/Write: R/W │ Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxx000000000110)

Bit	Reset	SW Default	Description
11:8	0x0	PCD1	PIXEL_CLK_DIVIDER: Pixel Clock Divider 0000= divide by 1 0001= divide by 1.5 0010= divide by 2 0011= divide by 3 0100= divide by 4 0101= divide by 6 0110= divide by 8 0111= divide by 9 1000= divide by 12 1001= divide by 16 1010= divide by 18 1011= divide by 24 1100= divide by 13 other= reserved 0 = PCD1 1 = PCD1H 2 = PCD2 3 = PCD3 4 = PCD4 5 = PCD6 6 = PCD8 7 = PCD9 8 = PCD12 9 = PCD16 10 = PCD18 11 = PCD24 12 = PCD13

Bit	Reset	SW Default	Description
7:0	0x6	NONE	SHIFT_CLK_DIVIDER: Shift Clock Divider. 0 = divide by 1 1 = divide by 1.5 2 = divide by 2 3 = divide by 2.5 4 = divide by 3 : : : : 254 = divide by 128 255 = divide by 128.5 Pixel clock divider is used to divide output of internal shift clock divider to generate internal pixel clock which is used to clock the internal horizontal and vertical counters. This divider also determines the output format for parallel interface, serial interface, and LCD SPI interface in conjunction with Display Data Format parameter. For 1-pixel/1-clock parallel display interface, valid settings are PCD1, PCD2, and PCD4. Note that the main reason to use PCD2 and PCD4 is to get higher frequency PP clock because the PP clock is always generated from the output of shift clock divider. For non 1-pixel/1-clock parallel display interface, valid settings are, PCD1H (2-pixel/3-clock), PCD2 (1-pixel/2-clock), and PCD3 (1-pixel/3-clock). For 1-channel serial display interface, valid settings are PCD3 (3-bpp 1-ch), PCD4 (3-bpp 1-ch), PCD6 (6-bpp 1-ch), PCD9 (9-bpp 1-ch), PCD12 (12-bpp 1-ch), PCD16 (16-bpp 1-ch), PCD18 (18-bpp 1-ch). For 2-channel serial display interface, valid settings are PCD2 (3-bpp 2-ch), PCD3 (6-bpp 2-ch), PCD6 (12-bpp 2-ch), PCD8 (16-bpp 2-ch), PCD9 (18-bpp 2-ch). For 3-channel serial display interface, valid settings are PCD1 (3-bpp 3-ch), PCD2 (6-bpp 3-ch), PCD3 (9-bpp 3-ch), PCD4 (12-bpp 3-ch), PCD6 (18-bpp 3-ch). For LCD SPI interface, valid settings are PCD12 (B4G4R4), PCD16 (B5G6R5), PCD18 (B6G6R6), PCD24 (B8G8R8), PCD8 (B5G6R5 with data/command bit), PCD6 (B5G6R5 with data/command start byte - depending on data/command bit), PCD4 (P8 for spi8), PCD9 (B5G6R5 with chip select deassertion at 8-bit boundary, spi16x2), PCD3 (P8 for spidc), PCD2 (B5G6R5 with data/command bit and chip select deassertion at 9-bit boundary, spi16x2dc), and PCD13 (spi12p2, no chip select deassertion between pairs of pixels).

24.8.37 DC_DISP_DISP_INTERFACE_CONTROL_0

Display Interface Control

This register specifies display interface options.

Note: *Only the PCD1 setting is allowed. Only default settings are allowed.*

Offset: 0x42f │ Byte Offset: 0x10bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0000)

Bit	Reset	Description
9	0x0	DISP_DATA_ORDER: Display Data Order. This is effective only for 1-pixel/2-clock 16-/18-/24- bit parallel interface 0= Red pixel is output in the first clock and blue pixel is output in the second cycle 1= Blue pixel is output in the first clock cycle and red pixel is output in the second clock cycle 0 = RED_BLUE 1 = BLUE_RED
8	0x0	DISP_DATA_ALIGNMENT: Display Data Alignment. This is effective for parallel display data format and the associated Initialization Sequence (IS). 0= Output data is MSB-aligned. For 1-pixel/1-clock parallel display, the output data ordering is the same regardless of display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 18-bpp so the 24-bit data ordering is: LD[5:0] are blue data bits 7-2 LD[11:6] are green data bits 7-2 LD[17:12] are red data bits 7-2 LD[19:18] are blue data bits 1-0 LD[21:20] are green data bits 1-0 LD[23:22] are red data bits 1-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 1= Output data is LSB-aligned. For 1-pixel/1-clock parallel display the output data ordering is determined by display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 24-bpp as follows: LD[7:0] are blue data bits 7-0 LD[15:8] are green data bits 7-0 LD[23:16] are red data bits 7-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 0 = MSB 1 = LSB
3:0	0x0	DISP_DATA_FORMAT: Display Data Format Pixel Clock Divider is used together with this parameter to determine the exact display data format. 0 = DF1P1C: 0= 1-pixel/1-clock up to 24-bit parallel 1 = DF1P2C24B: 1= 1-pixel/2-clock 24-bit parallel 2 = DF1P2C18B: 2= 1-pixel/2-clock 18-bit parallel or 2-pixel/3-clock 12-bit parallel or 1-pixel/3-clock 18-bit parallel NOTE: for 2-pixel/3-clock 12-bit parallel, the horizontal display active time must be even number of pixels. 3 = DF1P2C16B: 3= 1-pixel/2-clock 16-bit parallel 4 = DF1S: 4= 1-channel serial NOTE: 1-/2-/3-channel serial display interface supported is a low-voltage differential serial interface. 5 = DF2S: 5= 2-channel serial 6 = DF3S: 6= 3-channel serial 7 = DFSPI: 7= SPI serial 8 = DF1P3C24B: 8= 1-pixel/3-clock 24-bit parallel 9 = DF2P1C18B: 9= 2-pixel/1-clock 18-bit parallel 10 = DFDUAL1P1C18B: 10= 1-pixel/1-clock 18-bit parallel (used for dual output)

24.8.38 DC_DISP_DISP_COLOR_CONTROL_0

Display Color Control

Offset: 0x430 │ Byte Offset: 0x10c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxx0000xxx0x0x0xx00xxxx00xx0000)

Bit	Reset	Description
27	0x0	Reserved
26	0x0	Reserved
25	0x0	Reserved
24	0x0	Reserved
20	DISABLE	CMU_ENABLE: Enable CMU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	NON_BASE_COLOR: Non Base Color 0= zeros 1= ones
17	X	BLANK_COLOR: Blank Color 0= zeros 1= ones Non Base Color applies to least significant color bits which are not part of base color and it has higher priority over Border Color but lower priority over Blank color.
16	0x0	DISP_COLOR_SWAP: Display Color Swap 0= RGB (normal) 1= BGR (red-blue reverse) 0 = RGB 1 = BGR
13:12	0x0	ORD_DITHER_ROTATION: Ordered Dither Frame Rotation. This parameter specifies the rotation frequency of the dither matrix in terms of number of frames. If programmed to 0, there is no dither matrix rotation. If programmed to N, where N is larger than 0, the dither matrix is rotated clockwise every N frame.
9:8	X	DITHER_CONTROL: Dither Control 00= dither disabled 01= reserved 10= ordered dither 11= temporal dithering Design Note: initial dither matrix (where d is 2 dither bits) d=00 d=01 d=10 d=11 --------- --------- --------- --------- 0 0 1 0 0 1 0 1 --------- --------- --------- --------- 0 0 0 0 1 0 1 1 --------- ---- Note: 0 in the matrix specifies no addition to base color 1 in the matrix specifies incrementation of base color (with saturation) 0 = DISABLE 2 = ORDERED 3 = TEMPORAL
7:6	0x0	TEMPORAL_DITHER_PHASE: Temporal dither LFSR phase control 2'b01: random_data <= 34'h3fffffff 2'b10: random_data <= 34'h155555555 2'b11: random_data <= 34'h2aaaaaaaa 2'b00: retains previous value Temporal dither supports only 8 to 6 bits dithering.
3:0	0x0	BASE_COLOR_SIZE: Display Base Color Size. This parameter determines the number of bits per color after dither. 0= 6 bits 1= 1 bit 2= 2 bits 3= 3 bits 4= 4 bits 5= 5 bits 6= 5 bits for R,B and 6 bits for G 7= 3 bits for R,G and 2 bits for B 8= 8 bits, this also forces dither to be disabled. This setting can be used to output 24-bit data in 1-pixel/clock parallel display data format. 0 = BASE666 1 = BASE111 2 = BASE222 3 = BASE333 4 = BASE444 5 = BASE555 6 = BASE565 7 = BASE332 8 = BASE888

24.8.39 DC_DISP_COLOR_KEY0_LOWER_0

Color Key 0 Lower Value

Color Key 0 and Color Key 1

Two ranges of color key are defined and they are common for all windows because it is expected that typically only one window will have the color key enabled. Because there are two sets of color key, it is possible to have 2 windows each using one color key set.

Usage of this color key is described in the Display Color Key and Blending class.

Offset: 0x436 | Byte Offset: 0x10d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY0_L_A: Color Key 0 Alpha (0xFF) Lower value
23:16	0x0	COLOR_KEY0_L_B: Color Key 0 Blue (U) Lower value
15:8	0x0	COLOR_KEY0_L_G: Color Key 0 Green (Y) Lower value
7:0	0x0	COLOR_KEY0_L_R: Color Key 0 Red (V) Lower value

24.8.40 DC_DISP_COLOR_KEY0_UPPER_0

Color Key 0 Upper Value

Offset: 0x437 | Byte Offset: 0x10dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY0_U_A: Color Key 0 Alpha (0xFF) Upper value
23:16	0x0	COLOR_KEY0_U_B: Color Key 0 Blue (U) Upper value
15:8	0x0	COLOR_KEY0_U_G: Color Key 0 Green (Y) Upper value
7:0	0x0	COLOR_KEY0_U_R: Color Key 0 Red (V) Upper value

24.8.41 DC_DISP_COLOR_KEY1_LOWER_0

Color Key 1 Lower Value

Offset: 0x438 | Byte Offset: 0x10e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY1_L_A: Color Key 1 Alpha (0xFF) Lower value
23:16	0x0	COLOR_KEY1_L_B: Color Key 1 Blue (U) Lower value
15:8	0x0	COLOR_KEY1_L_G: Color Key 1 Green (Y) Lower value
7:0	0x0	COLOR_KEY1_L_R: Color Key 1 Red (V) Lower value

24.8.42 DC_DISP_COLOR_KEY1_UPPER_0

Color Key 1 Upper Value

Offset: 0x439 | Byte Offset: 0x10e4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	COLOR_KEY1_U_A: Color Key 1 Alpha (0xFF) Upper value
23:16	0x0	COLOR_KEY1_U_B: Color Key 1 Blue (U) Upper value
15:8	0x0	COLOR_KEY1_U_G: Color Key 1 Green (Y) Upper value
7:0	0x0	COLOR_KEY1_U_R: Color Key 1 Red (V) Upper value

24.8.43 DC_DISP_CURSOR_FOREGROUND_0

Cursor Foreground color

Class: Hardware Cursor

Hardware cursor is supported for 32x32 or for 64x64 2-bpp cursor.

Cursor start address is aligned to 1 KB boundary. All cursor registers except for cursor foreground and background colors are triple buffered.

GENERAL_UPDATE controls ASSEMBLY->ARM latching, GENERAL_ACT_REQ controls ARM->ACTIVE latching.

Cursor scaling and flipping are not implemented so this must be done by software if needed. Cursor H/V positions are signed number with respect to one of the display windows or with respect to upper left position of display active area as specified by cursor clipping parameter which also determines cursor clipping boundary. If cursor position is with respect to one of the display window and the corresponding display window is disabled then cursor will also be disabled.

In legacy cursor mode, only 32x32 and 64x64 are supported

Offset: 0x43c │ Byte Offset: 0x10f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CURSOR_FOREGROUND_B: Cursor Blue Foreground Color
15:8	X	CURSOR_FOREGROUND_G: Cursor Green Foreground Color
7:0	X	CURSOR_FOREGROUND_R: Cursor Red Foreground Color

24.8.44 DC_DISP_CURSOR_BACKGROUND_0

Cursor Background Color

Offset: 0x43d │ Byte Offset: 0x10f4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CURSOR_BACKGROUND_B: Cursor Blue Background Color
15:8	X	CURSOR_BACKGROUND_G: Cursor Green Background Color
7:0	X	CURSOR_BACKGROUND_R: Cursor Red Background Color

24.8.45 DC_DISP_CURSOR_START_ADDR_0

Cursor Start Address

Offset: 0x43e │ Byte Offset: 0x10f8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:28	X	CURSOR_CLIPPING: Cursor Clipping Select 0 = DISPLAY: 00= display 1 = WA: 01= window A 2 = WB: 10= window B 3 = WC: 11= window C
25: 24	X	CURSOR_SIZE: Cursor Size 0 = C32X32 1 = C64X64 2 = C128X128 3 = C256X256
21:0	X	CURSOR_START_ADDR: Cursor Start Address bits 31:10

24.8.46 DC_DISP_CURSOR_POSITION_0

Cursor Position

Cursor position is with respect to top-left corner of display active area, window A, window B, or window C as specified in the cursor clipping parameter.

Offset: 0x440 │ Byte Offset: 0x1100 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION: V cursor position (signed)
13:0	X	H_CURSOR_POSITION: H cursor position (signed)

24.8.47 DC_DISP_DC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register (where applicable) and Clock Gating Control

Note: *The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility. The clock override/ovr_mode fields of this register control the second level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.*

A '1' written to the rclk/wclk override filed will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to the legacy mode of operation (where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x480 | Byte Offset: 0x1200 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	DC_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	DC_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	DC_CCLK_OVERRIDE
17	0x0	DC_RCLK_OVERRIDE
16	0x0	DC_WCLK_OVERRIDE
3	DISABLE	DC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	DC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	DC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	DC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

24.8.48 DC_DISP_MCCIF_DISPLAY0A_HYST_0

Memory Client Hysteresis Control Register

Note: *Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.*

Offset: 0x481 | Byte Offset: 0x1204 | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0A2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0A2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0A2MC_HYST_TM
23:16	0x38	CSR_DISPLAY0A2MC_DHYST_TH

Bit	Reset	Description
15:8	0x10	CSR_DISPLAY0A2MC_DHYST_TM
7:0	0x58	CSR_DISPLAY0A2MC_HYST_REQ_TM

24.8.49 DC_DISP_MCCIF_DISPLAY0B_HYST_0

Memory Client Hysteresis Control Register

Note: *Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.*

Offset: 0x482 | Byte Offset: 0x1208 | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0B2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0B2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0B2MC_HYST_TM
23:16	0x40	CSR_DISPLAY0B2MC_DHYST_TH
15:8	0x1f	CSR_DISPLAY0B2MC_DHYST_TM
7:0	0x1f	CSR_DISPLAY0B2MC_HYST_REQ_TM

24.8.50 DC_DISP_MCCIF_DISPLAY0C_HYST_0

Memory Client Hysteresis Control Register

Note: *Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.*

Offset: 0x483 | Byte Offset: 0x120c | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0C2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0C2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0C2MC_HYST_TM
23:16	0x40	CSR_DISPLAY0C2MC_DHYST_TH
15:8	0x1f	CSR_DISPLAY0C2MC_DHYST_TM
7:0	0x1f	CSR_DISPLAY0C2MC_HYST_REQ_TM

24.8.51 DC_DISP_DISP_MISC_CONTROL_0

Miscellaneous Controls

Offset: 0x4c1 | Byte Offset: 0x1304 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx10)

Bit	Reset	Description
1	0x1	UF_LINE_FLUSH: Enable underflow line flush as opposed to end-of-frame flush. underflow line flush 0 = DISABLE; 1 = ENABLE

Bit	Reset	Description
0	0x0	PHASE_SHIFT_2P1C18B: Enable phase shift for 2P1C format phase shift SC0/SC1 will be delayed for one pixel clock cycle. In 2P1C format, data will hold for 2 pixel clocks, so either choice should work 0 = DISABLE 1 = ENABLE

24.8.52 DC_DISP_SD_CONTROL_0

The Smart Dimmer (PRISM2) function takes advantage of the fact that a perceived pixel brightness in an LCD depends on both the pixel brightness value and the backlight intensity to reduce the backlight intensity to save power. Statistics are gathered on the current video frame and an "enhancement" is applied to subsequent frames that increases the pixel brightness value and reduces the backlight brightness to give an overall image intensity that is mostly the same as before.

PRISM2 Control

Offset: 0x4c2 | Byte Offset: 0x1308 | Read/Write: R/W | Reset: 0x00004000 (0bx000xxxxxxxxxxx0100000000000000)

Bit	Reset	Description
30:29	BIAS0	K_INIT_BIAS: BIAS of the initial K bias MSB of count 0 = BIAS0: bias 0 1 = BIAS1: bias 1.0 2 = BIAS_HALF: bias 0.5 3 = BIAS_MSB
28	VSYNC	SD_FRAME_PROC_CONTROL: when to run per-frame processing. VSYNC is legacy behavior. 0 = VSYNC 1 = VPULSE2
15	DISABLE	SMOOTH_K_ENABLE: When enable, maximum raw K change per frame is limited to SMOOTH_K_INCR 0 = DISABLE 1 = ENABLE
14	ENABLE	SOFT_CLIPPING_ENABLE: When enabled, enhancement gain (K) is reduced for pixels above SOFT_CLIPPING_THRESHOLD level to avoid saturation (clipping). 0 = DISABLE 1 = ENABLE
13	DISABLE	SD_WINDOW_ENABLE: When enabled, constrain histogram (and therefore backlight) to a rectangular subset of display. The rectangle has upper/left corner described by SD_WINDOW_POSITION, and width/height by SD_WINDOW_SIZE. Useful for when display has regions that should not influence SD, such as letterbox borders, etc. Enhancement is always performed on whole display, regardless. DISABLE: SD histogram done over whole display ENABLE: SD histogram done over rectangular subset given by SD_WINDOW* registers 0 = DISABLE 1 = ENABLE
12	DISABLE	K_LIMIT_ENABLE: When enabled, Max. K is taken from K_LIMIT register rather than computed from AGGRESSIVENESS. DISABLE: K is limited by AGGRESSIVENESS ENABLE: K is limited by K_LIMIT register. 0 = DISABLE 1 = ENABLE
11	0x0	SD_CORRECTION_MODE: Determines which K values are used to modify the pixel values. MANUAL: The K values in the SD_MAN_K_VALUES register are used to modify pixel values. 0 = AUTO_CORRECT: AUTO_CORRECT: SD Hardware computed K values are used to 1 = MANUAL
10	0x0	SD_ONE_SHOT: Enables the PRISM2 function for one frame only. on which the SD statistics are gathered. NOTE: The SD_ENABLE field (see above) must be set to ONE_SHOT in order to use this function. 0 = DISABLE: Automatically cleared to DISABLE at the end of the frame 1 = ENABLE

Bit	Reset	Description
9:8	0x0	HW_UPDATE_DLY: Determines the delay - in video frames - of the update of the hardware enhancement value that is applied to the pixels. This is useful for allowing the software some time to update the backlight control, when the control must be sent via side-band control packets or by some other means of control that incurs a sizeable delay. Being able to delay the hardware update ensures that the modification of the pixels occurs as nearly simultaneously with the update of the backlight as possible. Value Description 0 No delay - pixels modified immediately 1 New enhancement value delayed by 1 frame. 2 New enhancement value delayed by 2 frames. 3 New enhancement value delayed by 3 frames.
7:5	0x0	AGGRESSIVENESS: The "aggressiveness" level of the PRISM2 algorithm. Higher aggressiveness levels result in higher power savings at the potential expense of image quality. The number programmed determines how many highlight pixels will be allowed to exceed the maximum representable brightness value and be clipped to that value. It also determines the maximum allowed enhancement value (k) applied to the pixel brightness. AGGRESSIVENESS Description Max pixels crushed (%) Max. k value 0 Essentially off 0% 1.00 1 Highest quality < 5% 1.10 2 Higher quality < 10% 1.15 3 Balanced < 15% 1.20 4 Higher battery life < 20% 1.25 5 Highest battery life < 25% 1.50
4:3	0x0	BIN_WIDTH: Width of the Histogram bins, in quantization levels. EIGHT = 8 levels per bin. Bins span range from 0 to 255 0 = ONE: ONE = 1 level per bin. Bins span range from 224 to 255 1 = TWO: TWO = 2 levels per bin. Bins span range from 192 to 255 2 = FOUR: FOUR = 4 levels per bin. Bins span range from 128 to 255 3 = EIGHT
2	0x0	USE_VID_LUMA: Use Video Luminance control of luminance: Luminance = MAX(R, G, B) ENABLE = use "video" luminance, which is determined by the coefficients in the SD_CSC_COEFFS register See the SD_CSC_COEFFS register for details. 0 = DISABLE: DISABLE = use Hue Saturation Value (HSV) version 1 = ENABLE
1:0	0x0	SD_ENABLE: Enables the PRISM2 Function. 0 = DISABLE: DISABLE = SD function disabled. 1 = ENABLE: ENABLE = SD function enabled and operational. 2 = ONE_SHOT = SD function enabled, but statistics gathering limited to the next frame only.

24.8.53 DC_DISP_SD_CSC_COEFF_0

Luminance calculation coefficients used to convert the red, green, and blue color components into a luminance value. The conversion is performed according to the following equation: Luminance = (R * R_COEFF + G * G_COEFF + B * B_COEFF) >> 4 It is suggested that the values of the coefficients be programmed as shown below, though user-defined color spaces are also accommodated. Color Space R_COEFF G_COEFF B_COEFF ITU-R Bt601 5 9 2 ITU-R Bt709 3 12 1. The coefficients do not have to be particularly accurate, hence their low precision, and the coefficients used are open to experimentation to obtain the best results.

Offset: 0x4c3 │ Byte Offset: 0x130c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx0000xxxx0000xxxx0000xxxx)

Bit	Reset	Description
23:20	0x0	B_COEFF: Blue coefficient for luminance calculation
15:12	0x0	G_COEFF: Green coefficient for luminance calculation
7:4	0x0	R_COEFF: Red coefficient for luminance calculation

24.8.54 DC_DISP_SD_LUT_0

Enhancement value (k) Look Up Table. Each LUT entry contains the value of k for each of the three color components. Since the value of k for the color components must be the reciprocal of the hardware-computed value of k, and the hardware value is guaranteed to be less than or equal to 1, the values in the LUT represent the fractional part of k, with an implied 1 to the left of the decimal place. For example, if the programmed value of R_LUT was 64 (01000000 in binary), then the actual value of k

generated would be 1.01000000 in binary or 1.25 in decimal. To program a default, linear response into the LUT, use the following code as a guide: for (i = 0; i < 9; i++) { t = (4096 / (8 + i)) - 256; if (t > 255) t = 255; R_LUT[i] = t; G_LUT[i] = t; B_LUT[i] = t; } For other non-linear response curves (for example, to take display gamma into consideration), this code will have to be modified.
This is an array of 9 identical register entries; the register fields below apply to each entry.

Offset: 0x4c4..0x4cc | Byte Offset: 0x1310..0x1330 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	B_LUT: Blue Enhancement value (k) Look Up Table
15:8	0x0	G_LUT: Green Enhancement value (k) Look Up Table
7:0	0x0	R_LUT: Red Enhancement value (k) Look Up Table

24.8.55 DC_DISP_SD_FLICKER_CONTROL_0

Flicker Reduction Control Register

The flicker control prevents rapid and frequent changes in the enhancement value.

Offset: 0x4cd | Byte Offset: 0x1334 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	THRESHOLD: The amount by which the currently calculated enhancement value must deviate from the currently active enhancement value for it to increment the TIME_LIMIT counter.
7:0	0x0	TIME_LIMIT: Length of time - in frames - that the enhancement value must deviate from the current value by more than THRESHOLD, before the enhancement value changes.

24.8.56 DC_DISP_SD_PIXEL_COUNT_0

Status / debug register showing the total number of active pixels

Offset: 0x4ce | Byte Offset: 0x1338 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	NUM_PIXELS: in the preceding output frame. Expressed as a quantity of 256 pixels. In other words, a 640 x 480 image has 307200 pixels. The value in this register would be 307200 / 256 = 1200

24.8.57 DC_DISP_SD_HISTOGRAM_0

Status/debug registers showing the gathered histogram data. Each register contains 4 histogram bins, for a total of 8 x 4 = 32 bins. Each bin has been approximately scaled to the number of pixels in the image so that a single quantization step in a bin represents a fraction of between 1/256 and 1/128 of the total number of pixels in the image.

This is an array of 8 identical register entries; the register fields below apply to each entry.

Offset: 0x4cf..0x4d6 | Byte Offset: 0x133c..0x1358 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	**X**	**BIN_3**
23:16	X	BIN_2
15:8	X	BIN_1
7:0	X	BIN_0

24.8.58 DC_DISP_SD_BL_PARAMETERS_0

Backlight response parameters. Defines the parameters for the backlight temporal response model.

Offset: 0x4d7 | Byte Offset: 0x135c | Read/Write: R/W | Reset: 0x00000400 (0bxxxxxxxx00000000xxxxx10000000000) | Default: 0x00ff0000

Bit	Reset	SW Default	Description
23:16	0x0	0xff	STEP: Determines the instantaneous portion of the target value of enhancement that is applied. 0 = 0%: response is entirely exponential and determined by TIME_CONSTANT 128 = 50%: response will instantly step up by 50% and will then be exponential. 255 = 100%: response is entirely instantaneous. TIME_CONSTANT has no effect.
10:0	0x400	NONE	TIME_CONSTANT: The time constant for the response curve. This value represents the fraction by which the value of enhancement value approaches the target value each frame. Example values are shown below: 0: The value will never reach the target (infinite TC) 512: The next value will be halfway between the current value and the target value. 1024: The next value will be 100% of the target value. In other words - an instantaneous response.

24.8.59 DC_DISP_SD_BL_TF_0

Backlight Transfer Function

Each register contains 4 points on the Transfer Function curve that defines how the backlight output changes with respect to the control input. Each point defines a value at the vertex of a 16 segment line. The 17th point is defined to be the maximum value (it is assumed 100% control == 100% light output).

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x4d8..0x4db | Byte Offset: 0x1360..0x136c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	POINT_3
23:16	X	POINT_2
15:8	X	POINT_1
7:0	X	POINT_0

24.8.60 DC_DISP_SD_BL_CONTROL_0

Backlight Control Register

Offset: 0x4dc | Byte Offset: 0x1370 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
15:8	RO	X	BRIGHTNESS: Backlight brightness modification value. This value is determined by the hardware according to all the other control registers and the image content. The amount by which the backlight should be modified is given as a fraction with 0 representing that the backlight should be off and 255 representing no change in the backlight intensity. Other values vary linearly between these two extremes. New BL control = (Old BL control * BRIGHTNESS) / 255

Bit	R/W	Reset	Description
1:0	RW	0x0	BL_MODE: Control Mode: and adjust the backlight brightness itself. PWM_AUTO: Hardware will adjust the backlight PWM control signal directly using the value in BRIGHTNESS. * OTHER VALUES ARE RESERVED FOR FUTURE USE * 0 = MANUAL: MANUAL: Hardware makes no BL corrections. Software must read the BRIGHTNESS field 1 = PWM_AUTO

24.8.61 DC_DISP_SD_HW_K_VALUES_0

Hardware-computed values of K for each color component. These values are used to modify the pixel values when CORRECTION_MODE in the SD_CONTROL register is set to AUTO_CORRECT. The numerical format is a fractional 10-bit number. Since we only ever want to make the pixel values bigger, there is an implied integer "1" not present in the register value. Only the fractional part is returned. Also in Tegra X1 devices, only the top 7 bits of the 10-bit K word have any meaning. The bottom 3 bits are always 0 and are reserved for future expansion. This register is read only.

Offset: 0x4dd │ Byte Offset: 0x1374 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	HW_K_BLUE: Value of K for blue pixels.
19:10	X	HW_K_GREEN: Value of K for green pixels.
9:0	X	HW_K_RED: Value of K for red pixels.

24.8.62 DC_DISP_SD_MAN_K_VALUES_0

Manual values of K for each color component. These values are used to modify the pixel values when CORRECTION_MODE in the SD_CONTROL register is set to MANUAL. The numerical format is a fractional 10-bit number. Since we only ever want to make the pixel values bigger, there is an implied integer "1" not present in the register value. Only the fractional part can be programmed. Therefore, K values can only be programmed from 1.0 to slightly less than 2.0.Also note that only the top 7 bits of the 10-bit K word have any effect in the hardware. The bottom 3 bits are reserved for future expansion.

Offset: 0x4de │ Byte Offset: 0x1378 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	MAN_K_BLUE: Value of K for blue pixels.
19:10	X	MAN_K_GREEN: Value of K for green pixels.
9:0	X	MAN_K_RED: Value of K for red pixels.

24.8.63 DC_DISP_SD_K_LIMIT_0

Offset: 0x4df │ Byte Offset: 0x137c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	K_LIMIT: When K_LIMIT_ENABLE=ENABLE, limits raw K independently of AGGRESSIVENESS. Currently, only the bottom 8 bits are effective. The upper 2 are reserved for future expansion.

24.8.64 DC_DISP_SD_WINDOW_POSITION_0

SD Window Position

Offset: 0x4e0 │ Byte Offset: 0x1380 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	SD_WIN_V_POSITION: SD window vertical position. This is specified with respect to the top edge of active display area.

Bit	Reset	Description
12:0	X	SD_WIN_H_POSITION: SD window horizontal position (pixels). This is specified with respect to the left edge of active display area.

24.8.65 DC_DISP_SD_WINDOW_SIZE_0

SD Window Size

Offset: 0x4e1 | Byte Offset: 0x1384 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	SD_WIN_V_SIZE: SD window vertical height (lines).
12:0	X	SD_WIN_H_SIZE: SD window horizontal width (pixels).

24.8.66 DC_DISP_SD_SOFT_CLIPPING_0

SD soft clipping parameters. Software programs both threshold and a reciprocal thereof.

Offset: 0x4e2 | Byte Offset: 0x1388 | Read/Write: R/W | Reset: 0x02000080 (0b00000010000000000xxxxxxx10000000)

Bit	Reset	Description
31:16	0x200	SOFT_CLIPPING_RECIP: Reciprocal of inverse threshold. Compute as RECIP = int(64*1024 * 1.0 / (256 - THRESHOLD)). For example, for THRESHOLD=128, RECIP = int(64*1024 * 1.0 / (256 - 128)) = 512.
7:0	0x80	SOFT_CLIPPING_THRESHOLD: Threshold at which pixel enhancement gain is reduced.

24.8.67 DC_DISP_SD_SMOOTH_K_0

Offset: 0x4e3 | Byte Offset: 0x138c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:0	X	SMOOTH_K_INCR: When SMOOTH_K_ENABLE=1, the raw K is changed at most by SMOOTH_K_INCR per frame. 8.6 fixed-point fraction, with resolution 1/64. Currently, only the top 12 bits are effective. The bottom 2 are reserved for future expansion, so the effective resolution is 1/16.

24.8.68 DC_DISP_BLEND_BACKGROUND_COLOR_0

Offset: 0x4e4 | Byte Offset: 0x1390 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	BKGND_ALPHA
23:16	0x0	BKGND_BLUE
15:8	0x0	BKGND_GREEN
7:0	0x0	BKGND_RED

24.8.69 DC_DISP_INTERLACE_CONTROL_0

1080i Related Register

This register controls whether interlacing is enabled or disabled. When interlacing is enabled, the data is fetched from memory at 1080i (interlaced) and all fields are passed directly to the panel. A set of even (normal) and odd (new for interlacing) registers define the field base addresses and timings.

If interlacing is disabled, the following interlacing related registers will have NO effect.

Offset: 0x4e5 | Byte Offset: 0x1394 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
2	RO	X	INTERLACE_STATUS: 0 = field1 1 = field2 0 = FIELD1 1 = FIELD2
1	RW	0x0	INTERLACE_START: 0 = field1 1 = field2 0 = FIELD1 1 = FIELD2
0	RW	0x0	INTERLACE_ENABLE: 0 = DISABLE 1 = ENABLE

24.8.70 DC_DISP_INTERLACE_FIELD2_REF_TO_SYNC_0

1080i Related Register

This register controls the ref to sync offset for H sync and V sync for FIELD2.

Offset: 0x4e6 | Byte Offset: 0x1398 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_REF_TO_SYNC: V reference to VSYNC for FIELD2. IMP - Units are in pixel clocks. Generally, for the non-interlaced timing, the units are in line clocks; however, for interlaced V sync must be generated at a pixel granularity with respect to V ref.
12:0	X	FIELD2_H_REF_TO_SYNC: H reference to HSYNC for FIELD2. Units are in pixel clocks.

24.8.71 DC_DISP_INTERLACE_FIELD2_SYNC_WIDTH_0

1080i Related Register

This register controls the H and V sync widths for FIELD2.

Offset: 0x4e7 | Byte Offset: 0x139c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_SYNC_WIDTH: This field controls the V sync widths for FIELD2. Units are in line clocks.
12:0	X	FIELD2_H_SYNC_WIDTH: This field controls the H sync widths for FIELD2. Units are in pixel clocks.

24.8.72 DC_DISP_INTERLACE_FIELD2_BACK_PORCH_0

1080i Related Register

This register controls the H and V back porch widths for FIELD2.

Offset: 0x4e8 | Byte Offset: 0x13a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_BACK_PORCH: V back porch. Units are in line clocks.
12:0	X	FIELD2_H_BACK_PORCH: H back porch. Units are in pixel clocks.

24.8.73 DC_DISP_INTERLACE_FIELD2_FRONT_PORCH_0

1080i Related Register

This register controls the H and V front porch widths for FIELD2.

Offset: 0x4e9 | Byte Offset: 0x13a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_FRONT_PORCH: V front porch. Units are in line clocks.
12:0	X	FIELD2_H_FRONT_PORCH: H front porch. Units are in pixel clocks.

24.8.74 DC_DISP_INTERLACE_FIELD2_DISP_ACTIVE_0

1080i Related Register

This register controls the H and V active widths for FIELD2.

Offset: 0x4ea | Byte Offset: 0x13a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIELD2_V_DISP_ACTIVE: V active width. Units are in line clocks.
12:0	X	FIELD2_H_DISP_ACTIVE: H active width. Units are in pixel clocks.

24.8.75 DC_DISP_CURSOR_UNDERFLOW_CTRL_0

Offset: 0x4eb | Byte Offset: 0x13ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
7	0x0	CURSOR_UFLOW_CYA: 0 = DISABLE 1 = ENABLE
0	0x0	CURSOR_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

24.8.76 DC_DISP_CURSOR_START_ADDR_HI_0

Cursor Start Address

Offset: 0x4ec | Byte Offset: 0x13b0 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	CURSOR_START_ADDR_HI: Cursor Start Address bits 33:32

24.8.77 DC_DISP_CURSOR_INTERLACE_CONTROL_0

This register controls whether interlacing is enabled or disabled. If interlacing is disabled, the interlacing related registers will have NO effect.

Cursor Start Address bits 33:32

Offset: 0x4ee | Byte Offset: 0x13b8 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00x00)

Bit	R/W	Reset	Description
4	RW	0x0	CURSOR_INTERLACE_FIELD2_VOFF_INCR: FIELD2 v position incr Enable 0 = DISABLE 1 = ENABLE
3	RW	0x0	CURSOR_INTERLACE_FIELD1_VOFF_INCR: FIELD1 v position incr Enable 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
2	RO	X	CURSOR_INTERLACE_STATUS: Current line start. 0= line 0 1= line 1 0 = LINE0 1 = LINE1
1	RW	0x0	CURSOR_INTERLACE_START: Starting line 0= line 0 1= line 1 0 = LINE0 1 = LINE1
0	RW	0x0	CURSOR_INTERLACE_ENABLE: Interlace enable 0 = DISABLE 1 = ENABLE

24.8.78 DC_DISP_CSC2_CONTROL_0

The CSC2 can be used for RGB to YCbCr conversion used in 1080i HDMI output. This hardcoded color space converter supports YCBCR709 and YCBCR601 output as YUV444.

Offset: 0x4ef | Byte Offset: 0x13bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	LIMIT_RGB_COLOR: scale RGB [0,255] to [16,235] 0 = DISABLE 1 = ENABLE
1:0	RGB	OUTPUT_COLOR_SELECT: output color select: RGB for normal output, YCBCR709 for HDMI or YCBCR601 for older television color formats. 0 = RGB 1 = YCBCR709 2 = YCBCR601

24.8.79 DC_DISP_BLEND_CURSOR_CONTROL_0

In the last blending stage, the cursor is blended with the output of the previous blending stage.

BLEND_CURSOR_CONTROL controls the blend behavior:

output = cursor_pixel_color * SRC_FACTOR + previous_stage_output * DST_FACTOR

Offset: 0x4f1 | Byte Offset: 0x13c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx0xxxxxx00xxxxxx0000000000)

Bit	Reset	Description
24	0x0	CURSOR_MODE_SELECT: LEGACY: LEGACY cursor is 2bpp (4 pixels in 1 byte). The output pixel is selected from: 0--CURSOR_BACKGROUND 1--CURSOR_FOREGROUND 2--Previous_stage_output 3--Inverted previous_stage_output CURSOR_SRC_BLEND_FACTOR_SELECT,CURSOR_DST_BLEND_FACTOR_SELECT and CURSOR_ALPHA are ignored in this mode. NORMAL: NORMAL cursor is 32bpp (T_R8G8B8A8). Cursor_pixel_color and cursor_pixel_alpha are read from memory. 0 = LEGACY 1 = NORMAL
17:16	0x0	CURSOR_DST_BLEND_FACTOR_SELECT. Controls the DST_FACTOR. ZERO = 0 K1 = CURSOR_ALPHA/0xFF NEG_K1_TIMES_SRC = 1 - (CURSOR_ALPHA/0xFF) * (cursor_pixel_alpha/0xFF) 0 = ZERO 1 = K1 2 = NEG_K1_TIMES_SRC

Bit	Reset	Description
9:8	0x0	CURSOR_SRC_BLEND_FACTOR_SELECT. Controls the SRC_FACTOR: K1 = CURSOR_ALPHA/0xFF K1_TIMES_SRC = (CURSOR_ALPHA/0xFF) * (cursor_pixel_alpha/0xFF) 0 = K1 1 = K1_TIMES_SRC
7:0	0x0	CURSOR_ALPHA

24.8.80 DC_DISP_DVFS_CURSOR_CONTROL_0

Offset: 0x4f2 | Byte Offset: 0x13c8 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000011)

Bit	Reset	Description
8	DISABLE	CURSOR_DVFS_ENABLE: DISABLE: Forces ready_for_latency_event true (i.e., always allows DVFS event) ENABLE: Allows ready_for_latency_event to toggle, based on CURSOR_DVFS_THRESHOLD 0 = DISABLE 1 = ENABLE
7:0	0x3	CURSOR_DVFS_THRESHOLD

24.8.81 DC_DISP_CURSOR_UFLOW_DBG_PIXEL_0

Offset: 0x4f3 | Byte Offset: 0x13cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CURSOR_UFLOW_DBG_PIXEL

24.8.82 DC_DISP_CURSOR_SPOOLUP_CONTROL_0

Programmable spool up for cursor. If spoolup count > hc_vactive_start, it will be clamped to zero.

Offset: 0x4f4 | Byte Offset: 0x13d0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000001)

Bit	Reset	Description
7:0	0x1	CURSOR_SPOOLUP_START

24.8.83 DC_DISP_DISPLAY_CLK_GATE_OVERRIDE_0

Offset: 0x4f5 | Byte Offset: 0x13d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CMU_CLK_GATE_OVERRIDE: Disable clock-gating of the CMU module. 0 = DISABLE 1 = ENABLE
0	0x0	CURSOR_CLK_GATE_OVERRIDE: Disable clock-gating of cursor memfetch/control modules 0 = DISABLE 1 = ENABLE

24.8.84 DC_DISP_DISPLAY_DBG_TIMING_0

Offset: 0x4f6 | Byte Offset: 0x13d8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	H_BLANK
28:16	X	H_COUNT

Bit	Reset	Description
15	X	V_BLANK
12:0	X	V_COUNT

24.8.85 DC_DISP_DISPLAY_SPARE0_0

Offset: 0x4f7 | Byte Offset: 0x13dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	**0x0**	**SCRATCH_0**
13:4	0x0	dsc_rc_overflow_thresh[9:0] Used to program the threshold of “bufferFullness + throttle_offset” (currently, it is -172),
3:2	0x0	dsc_rc_solution_mode[1:0] 0: disable rc update 1: use solution#2 rc fix 2. Use solution#3 rc fix.
1	0x0	dsc_check_flatness2: ((dsc_flatness_fix_en == 0) || (pps->slice_width%3 != 1))? 1: 0
0	0x0	dsc_flatness_fix_en: Diagnostic bit for flatness updates only

24.8.86 DC_DISP_DISPLAY_SPARE1_0

Offset: 0x4f8 | Byte Offset: 0x13e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_1

24.9 WINC_AD/BD/CD Registers

These registers control windows A, B and C parameters.

Notes:

- *There are three copies of these registers for windows A, B, and C. Windows D and T have subsets of these registers.*
- *In the register and field names, <AD/BD/CD> indicates window A, B, or C depending on the window selection. Window selection is determined by which Window Direct address range is used.*

24.9.1 DC_<A/B/C>_WINC_<AD/BD/CD>_COLOR_PALETTE_0

This is used for palletized data format (color depth of 8 bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp).

Each window has its own color palette which consists of three 256x8 tables which can be written by the host and indexed (read) by the window.

For palletized data formats less than 8 bpp, the pixel data is aligned to least significant bits of the palette index (address), and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3:0 of the palette index, and bits 7:4 of the palette index are set to bits 7:4 of the Palette Color Extension.

Note that a host read is assumed to be not needed - software can cache the color palette in system memory.

This is an array of 256 identical register entries; the register fields below apply to each entry.

Window AD/BD/CD Color Palette

AD: Offset: 0xa00..0xaff | Byte Offset: 0x2800..0x2bfc | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xc00..0xcff | Byte Offset: 0x3000..0x33fc | Read/Write: WO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xe00..0xeff | Byte Offset: 0x3800..0x3bfc | Read/Write: WO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	<AD/BD/CD>_COLOR_PALETTE_B: Blue Color Palette
15:8	X	<AD/BD/CD>_COLOR_PALETTE_G: Green Color Palette
7:0	X	<AD/BD/CD>_COLOR_PALETTE_R: Red Color Palette

24.9.2 DC_<A/B/C>_WINC_<AD/BD/CD>_PALETTE_COLOR_EXT_0

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette.

Window AD/BD/CD Palette Color Extension

AD: Offset: 0xb00 | Byte Offset: 0x2c00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd00 | Byte Offset: 0x3400 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf00 | Byte Offset: 0x3c00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:1	X	<AD/BD/CD>_PALETTE_COLOR_EXT: Window A/B/C Palette Color Extension. Bits 7:1 are used for 1-bpp mode, bits 7:2 are used for 2-bpp mode, and bits 7:4 are used for 4-bpp mode

24.9.3 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P00_0

Horizontal Scaling Filter Coefficients

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed values ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed values ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned values ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6 pixels and
- Coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

The sum of all coefficients for each phase should be 128 typically, and software should never program the sum of all coefficients for a phase to be more than 128.

For each horizontal positional phase, the 6 filter coefficients require 32 register bits.

Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window AD/BD/CD Horizontal Filter phase 00

AD: Offset: 0xb01 | Byte Offset: 0x2c04 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd01 | Byte Offset: 0x3404 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf01 | Byte Offset: 0x3c04 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)

Bit	Reset	Description
28:24	X	<AD/BD/CD>_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	<AD/BD/CD>_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	<AD/BD/CD>_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	<AD/BD/CD>_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	<AD/BD/CD>_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

24.9.4 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P01_0

Window AD/BD/CD Horizontal Filter phase 01

AD: Offset: 0xb02 | Byte Offset: 0x2c08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd02 | Byte Offset: 0x3408 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf02 | Byte Offset: 0x3c08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	<AD/BD/CD>_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	<AD/BD/CD>_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	<AD/BD/CD>_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	<AD/BD/CD>_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	<AD/BD/CD>_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

24.9.5 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P02_0

Window AD/BD/CD Horizontal Filter phase 02

AD: Offset: 0xb03 | Byte Offset: 0x2c0c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd03 | Byte Offset: 0x340c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf03 | Byte Offset: 0x3c0c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	<AD/BD/CD>_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	<AD/BD/CD>_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	<AD/BD/CD>_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)
7:3	X	<AD/BD/CD>_H_FILTER_P02C1: Phase 02 coefficient 1 (typically -8)
2:0	X	<AD/BD/CD>_H_FILTER_P02C0: Phase 02 coefficient 0 (typically 1)

24.9.6 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P03_0

Window AD/BD/CD Horizontal Filter phase 03

AD: Offset: 0xb04 | Byte Offset: 0x2c10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd04 | Byte Offset: 0x3410 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf04 | Byte Offset: 0x3c10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	<AD/BD/CD>_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	<AD/BD/CD>_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	<AD/BD/CD>_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	<AD/BD/CD>_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)

Bit	Reset	Description
2:0	X	<AD/BD/CD>_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

24.9.7 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P04_0

Window AD/BD/CD Horizontal Filter phase 04

AD: Offset: 0xb05 | Byte Offset: 0x2c14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd05 | Byte Offset: 0x3414 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf05 | Byte Offset: 0x3c14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	<AD/BD/CD>_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	<AD/BD/CD>_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	<AD/BD/CD>_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	<AD/BD/CD>_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	<AD/BD/CD>_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

24.9.8 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P05_0

Window AD/BD/CD Horizontal Filter phase 05

AD: Offset: 0xb06 | Byte Offset: 0x2c18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd06 | Byte Offset: 0x3418 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf06 | Byte Offset: 0x3c18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	**X**	**<AD/BD/CD>_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)**
28:24	X	<AD/BD/CD>_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	<AD/BD/CD>_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)
15:8	X	<AD/BD/CD>_H_FILTER_P05C2: Phase 05 coefficient 2 (typically 102)
7:3	X	<AD/BD/CD>_H_FILTER_P05C1: Phase 05 coefficient 1 (typically -15)
2:0	X	<AD/BD/CD>_H_FILTER_P05C0: Phase 05 coefficient 0 (typically 3)

24.9.9 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P06_0

Window AD/BD/CD Horizontal Filter phase 06

AD: Offset: 0xb07 | Byte Offset: 0x2c1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd07 | Byte Offset: 0x341c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf07 | Byte Offset: 0x3c1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	**X**	**<AD/BD/CD>_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)**
28:24	X	<AD/BD/CD>_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	<AD/BD/CD>_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	<AD/BD/CD>_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	<AD/BD/CD>_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	<AD/BD/CD>_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

24.9.10 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P07_0

Window AD/BD/CD Horizontal Filter phase 07

AD: Offset: 0xb08 | Byte Offset: 0x2c20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd08 | Byte Offset: 0x3420 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf08 | Byte Offset: 0x3c20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	<AD/BD/CD>_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	<AD/BD/CD>_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	<AD/BD/CD>_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	<AD/BD/CD>_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	<AD/BD/CD>_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

24.9.11 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P08_0

Window AD/BD/CD Horizontal Filter phase 08

AD: Offset: 0xb09 | Byte Offset: 0x2c24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd09 | Byte Offset: 0x3424 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf09 | Byte Offset: 0x3c24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	<AD/BD/CD>_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)
23:16	X	<AD/BD/CD>_H_FILTER_P08C3: Phase 08 coefficient 3 (typically 76)
15:8	X	<AD/BD/CD>_H_FILTER_P08C2: Phase 08 coefficient 2 (typically 76)
7:3	X	<AD/BD/CD>_H_FILTER_P08C1: Phase 08 coefficient 1 (typically -15)
2:0	X	<AD/BD/CD>_H_FILTER_P08C0: Phase 08 coefficient 0 (typically 3)

24.9.12 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P09_0

Window AD/BD/CD Horizontal Filter phase 09

AD: Offset: 0xb0a | Byte Offset: 0x2c28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd0a | Byte Offset: 0x3428 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf0a | Byte Offset: 0x3c28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	<AD/BD/CD>_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	<AD/BD/CD>_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	<AD/BD/CD>_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	<AD/BD/CD>_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	<AD/BD/CD>_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

24.9.13 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P0A_0

Window AD/BD/CD Horizontal Filter phase 0A

AD: Offset: 0xb0b | Byte Offset: 0x2c2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd0b | Byte Offset: 0x342c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf0b | Byte Offset: 0x3c2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	<AD/BD/CD>_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	<AD/BD/CD>_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	<AD/BD/CD>_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	<AD/BD/CD>_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	<AD/BD/CD>_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

24.9.14 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P0B_0

Window AD/BD/CD Horizontal Filter phase 0B

AD: Offset: 0xb0c | Byte Offset: 0x2c30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd0c | Byte Offset: 0x3430 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf0c | Byte Offset: 0x3c30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
28:24	X	<AD/BD/CD>_H_FILTER_P0BC4: Phase 0B coefficient 4 (typically -15)
23:16	X	<AD/BD/CD>_H_FILTER_P0BC3: Phase 0B coefficient 3 (typically 102)
15:8	X	<AD/BD/CD>_H_FILTER_P0BC2: Phase 0B coefficient 2 (typically 47)
7:3	X	<AD/BD/CD>_H_FILTER_P0BC1: Phase 0B coefficient 1 (typically -11)
2:0	X	<AD/BD/CD>_H_FILTER_P0BC0: Phase 0B coefficient 0 (typically 2)

24.9.15 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P0C_0

Window AD/BD/CD Horizontal Filter phase 0C

AD: Offset: 0xb0d | Byte Offset: 0x2c34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd0d | Byte Offset: 0x3434 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf0d | Byte Offset: 0x3c34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	<AD/BD/CD>_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	<AD/BD/CD>_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	<AD/BD/CD>_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	<AD/BD/CD>_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	<AD/BD/CD>_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

24.9.16 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P0D_0

Window AD/BD/CD Horizontal Filter phase 0D

AD: Offset: 0xb0e | Byte Offset: 0x2c38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd0e | Byte Offset: 0x3438 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf0e | Byte Offset: 0x3c38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	<AD/BD/CD>_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	<AD/BD/CD>_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	<AD/BD/CD>_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)

Bit	Reset	Description
7:3	X	<AD/BD/CD>_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	<AD/BD/CD>_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

24.9.17 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P0E_0

Window AD/BD/CD Horizontal Filter phase 0E

AD: Offset: 0xb0f | Byte Offset: 0x2c3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd0f | Byte Offset: 0x343c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf0f | Byte Offset: 0x3c3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	<AD/BD/CD>_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	<AD/BD/CD>_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	<AD/BD/CD>_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	<AD/BD/CD>_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	<AD/BD/CD>_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

24.9.18 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_P0F_0

Window AD/BD/CD Horizontal Filter phase 0F

AD: Offset: 0xb10 | Byte Offset: 0x2c40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd10 | Byte Offset: 0x3440 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf10 | Byte Offset: 0x3c40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	<AD/BD/CD>_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	<AD/BD/CD>_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	<AD/BD/CD>_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	<AD/BD/CD>_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	<AD/BD/CD>_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	<AD/BD/CD>_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

24.9.19 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_YOF_0

Color Space Conversion Coefficients

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control. The CSC can only be enabled for window AD/BD/CD controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = 1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, and KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B, correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window AD/BD/CD CSC Y Offset

AD: Offset: 0xb11 | Byte Offset: 0x2c44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd11 | Byte Offset: 0x3444 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf11 | Byte Offset: 0x3c44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_CSC_YOF: Y Offset in s.7.0 format

24.9.20 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_KYRGB_0

Window AD/BD/CD CSC Y Coefficient (gain) for RGB

AD: Offset: 0xb12 | Byte Offset: 0x2c48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd12 | Byte Offset: 0x3448 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf12 | Byte Offset: 0x3c48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	<AD/BD/CD>_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

24.9.21 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_KUR_0

Window AD/BD/CD CSC U coefficient for R

AD: Offset: 0xb13 | Byte Offset: 0x2c4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd13 | Byte Offset: 0x344c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf13 | Byte Offset: 0x3c4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	<AD/BD/CD>_CSC_KUR: U coefficients for R in s.2.8 format

24.9.22 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_KVR_0

Window AD/BD/CD CSC V coefficient for R

AD: Offset: 0xb14 | Byte Offset: 0x2c50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd14 | Byte Offset: 0x3450 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf14 | Byte Offset: 0x3c50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	**X**	**<AD/BD/CD>_CSC_KVR: V coefficients for R in s.2.8 format**

24.9.23 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_KUG_0

Window AD/BD/CD CSC U coefficient for G

AD: Offset: 0xb15 | Byte Offset: 0x2c54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd15 | Byte Offset: 0x3454 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf15 | Byte Offset: 0x3c54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	<AD/BD/CD>_CSC_KUG: U coefficients for G in s.1.8 format

24.9.24 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_KVG_0

Window AD/BD/CD CSC V coefficient for G

AD: Offset: 0xb16 | Byte Offset: 0x2c58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd16 | Byte Offset: 0x3458 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf16 | Byte Offset: 0x3c58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	<AD/BD/CD>_CSC_KVG: V coefficients for G in s.1.8 format

24.9.25 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_KUB_0

Window AD/BD/CD CSC U coefficient for B

AD: Offset: 0xb17 | Byte Offset: 0x2c5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd17 | Byte Offset: 0x345c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf17 | Byte Offset: 0x3c5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	<AD/BD/CD>_CSC_KUB: U coefficients for B in s.2.8 format

24.9.26 DC_<A/B/C>_WINC_<AD/BD/CD>_CSC_KVB_0

Window AD/BD/CD CSC V coefficient for B

AD: Offset: 0xb18 | Byte Offset: 0x2c60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd18 | Byte Offset: 0x3460 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf18 | Byte Offset: 0x3c60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	<AD/BD/CD>_CSC_KVB: V coefficients for B in s.2.8 format

24.9.27 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P00_0

Vertical Scaling Filter Coefficients

The vertical scaling filter is a 2-tap filter with 4-bit positional phase. Coefficients 0 and 1 are 8-bit unsigned values ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2 pixels, and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

The sum of all coefficients for each phase should be 128 typically. Therefore coefficient 1 can be calculated from (1 - coefficient 0) and only coefficient 0 is programmed. For each vertical positional phase, the filter coefficient requires 8 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window AD/BD/CD Vertical Filter phase 00

AD: Offset: 0xb19 | Byte Offset: 0x2c64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd19 | Byte Offset: 0x3464 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf19 | Byte Offset: 0x3c64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

24.9.28 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P01_0

Window AD/BD/CD Vertical Filter phase 01

AD: Offset: 0xb1a | Byte Offset: 0x2c68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd1a | Byte Offset: 0x3468 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf1a | Byte Offset: 0x3c68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

24.9.29 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P02_0

Window AD/BD/CD Vertical Filter phase 02

AD: Offset: 0xb1b | Byte Offset: 0x2c6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd1b | Byte Offset: 0x346c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf1b | Byte Offset: 0x3c6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

24.9.30 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P03_0

Window AD/BD/CD Vertical Filter phase 03

AD: Offset: 0xb1c | Byte Offset: 0x2c70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd1c | Byte Offset: 0x3470 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf1c | Byte Offset: 0x3c70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

24.9.31 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P04_0

Window AD/BD/CD Vertical Filter phase 04

AD: Offset: 0xb1d | Byte Offset: 0x2c74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd1d | Byte Offset: 0x3474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf1d | Byte Offset: 0x3c74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

24.9.32 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P05_0

Window AD/BD/CD Vertical Filter phase 05

AD: Offset: 0xb1e | Byte Offset: 0x2c78 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd1e | Byte Offset: 0x3478 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf1e | Byte Offset: 0x3c78 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

24.9.33 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P06_0

Window AD/BD/CD Vertical Filter phase 06

AD: Offset: 0xb1f | Byte Offset: 0x2c7c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd1f | Byte Offset: 0x347c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf1f | Byte Offset: 0x3c7c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

24.9.34 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P07_0

Window AD/BD/CD Vertical Filter phase 07

AD: Offset: 0xb20 | Byte Offset: 0x2c80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd20 | Byte Offset: 0x3480 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf20 | Byte Offset: 0x3c80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

24.9.35 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P08_0

Window AD/BD/CD Vertical Filter phase 08

AD: Offset: 0xb21 | Byte Offset: 0x2c84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd21 | Byte Offset: 0x3484 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf21 | Byte Offset: 0x3c84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

24.9.36 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P09_0

Window AD/BD/CD Vertical Filter phase 09

AD: Offset: 0xb22 | Byte Offset: 0x2c88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd22 | Byte Offset: 0x3488 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf22 | Byte Offset: 0x3c88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

24.9.37 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P0A_0

Window AD/BD/CD Vertical Filter phase 0A

AD: Offset: 0xb23 | Byte Offset: 0x2c8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd23 | Byte Offset: 0x348c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf23 | Byte Offset: 0x3c8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

24.9.38 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P0B_0

Window AD/BD/CD Vertical Filter phase 0B

AD: Offset: 0xb24 | Byte Offset: 0x2c90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd24 | Byte Offset: 0x3490 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf24 | Byte Offset: 0x3c90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

24.9.39 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P0C_0

Window AD/BD/CD Vertical Filter phase 0C

AD: Offset: 0xb25 | Byte Offset: 0x2c94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd25 | Byte Offset: 0x3494 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf25 | Byte Offset: 0x3c94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

24.9.40 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P0D_0

Window AD/BD/CD Vertical Filter phase 0D

AD: Offset: 0xb26 | Byte Offset: 0x2c98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd26 | Byte Offset: 0x3498 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf26 | Byte Offset: 0x3c98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

24.9.41 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P0E_0

Window AD/BD/CD Vertical Filter phase 0E

AD: Offset: 0xb27 | Byte Offset: 0x2c9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd27 | Byte Offset: 0x349c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf27 | Byte Offset: 0x3c9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

24.9.42 DC_<A/B/C>_WINC_<AD/BD/CD>_V_FILTER_P0F_0

Window AD/BD/CD Vertical Filter phase 0F

AD: Offset: 0xb28 | Byte Offset: 0x2ca0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd28 | Byte Offset: 0x34a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf28 | Byte Offset: 0x3ca0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	<AD/BD/CD>_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

24.9.43 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P00_0

Window AD/BD/CD Horizontal Filter phase 00

AD: Offset: 0xb29 | Byte Offset: 0x2ca4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd29 | Byte Offset: 0x34a4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf29 | Byte Offset: 0x3ca4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P00C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P00C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P00C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P00C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P00C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P00C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.44 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P01_0

Window AD/BD/CD Horizontal Filter phase 01

AD: Offset: 0xb2a | Byte Offset: 0x2ca8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd2a | Byte Offset: 0x34a8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf2a | Byte Offset: 0x3ca8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P01C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P01C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P01C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P01C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P01C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P01C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.45 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P02_0

Window AD/BD/CD Horizontal Filter phase 02

AD: Offset: 0xb2b | Byte Offset: 0x2cac | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd2b | Byte Offset: 0x34ac | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf2b | Byte Offset: 0x3cac | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P02C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P02C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P02C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P02C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P02C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P02C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.46 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P03_0

Window AD/BD/CD Horizontal Filter phase 03

AD: Offset: 0xb2c | Byte Offset: 0x2cb0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd2c | Byte Offset: 0x34b0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf2c | Byte Offset: 0x3cb0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P03C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P03C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P03C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P03C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P03C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P03C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.47 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P04_0

Window AD/BD/CD Horizontal Filter phase 04

AD: Offset: 0xb2d | Byte Offset: 0x2cb4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd2d | Byte Offset: 0x34b4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf2d | Byte Offset: 0x3cb4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P04C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P04C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P04C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P04C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P04C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P04C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.48 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P05_0

Window AD/BD/CD Horizontal Filter phase 05

AD: Offset: 0xb2e | Byte Offset: 0x2cb8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd2e | Byte Offset: 0x34b8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf2e | Byte Offset: 0x3cb8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P05C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P05C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P05C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P05C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P05C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P05C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.49 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P06_0

Window AD/BD/CD Horizontal Filter phase 06

AD: Offset: 0xb2f | Byte Offset: 0x2cbc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd2f | Byte Offset: 0x34bc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf2f | Byte Offset: 0x3cbc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P06C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P06C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P06C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P06C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P06C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P06C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.50 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P07_0

Window AD/BD/CD Horizontal Filter phase 07

AD: Offset: 0xb30 | Byte Offset: 0x2cc0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd30 | Byte Offset: 0x34c0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf30 | Byte Offset: 0x3cc0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P07C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P07C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P07C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P07C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P07C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
1:0	X	<AD/BD/CD>_H_FILTER_HI_P07C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.51 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P08_0

Window AD/BD/CD Horizontal Filter phase 08

AD: Offset: 0xb31 | Byte Offset: 0x2cc4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd31 | Byte Offset: 0x34c4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf31 | Byte Offset: 0x3cc4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P08C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P08C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P08C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P08C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P08C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P08C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.52 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P09_0

Window AD/BD/CD Horizontal Filter phase 09

AD: Offset: 0xb32 | Byte Offset: 0x2cc8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd32 | Byte Offset: 0x34c8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf32 | Byte Offset: 0x3cc8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P09C5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P09C4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P09C3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P09C2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P09C1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P09C0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.53 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P0A_0

Window AD/BD/CD Horizontal Filter phase 0A

AD: Offset: 0xb33 | Byte Offset: 0x2ccc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd33 | Byte Offset: 0x34cc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf33 | Byte Offset: 0x3ccc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P0AC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P0AC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.

Bit	Reset	Description
5	X	<AD/BD/CD>_H_FILTER_HI_P0AC3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P0AC2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P0AC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P0AC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.54 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P0B_0

Window AD/BD/CD Horizontal Filter phase 0B

AD: Offset: 0xb34 | Byte Offset: 0x2cd0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd34 | Byte Offset: 0x34d0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf34 | Byte Offset: 0x3cd0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P0BC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P0BC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P0BC3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P0BC2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P0BC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P0BC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.55 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P0C_0

Window AD/BD/CD Horizontal Filter phase 0C

AD: Offset: 0xb35 | Byte Offset: 0x2cd4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd35 | Byte Offset: 0x34d4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf35 | Byte Offset: 0x3cd4 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P0CC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P0CC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P0CC3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P0CC2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P0CC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P0CC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.56 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P0D_0

Window AD/BD/CD Horizontal Filter phase 0D

AD: Offset: 0xb36 | Byte Offset: 0x2cd8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd36 | Byte Offset: 0x34d8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf36 | Byte Offset: 0x3cd8 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P0DC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P0DC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P0DC3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P0DC2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P0DC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P0DC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.57 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P0E_0

Window AD/BD/CD Horizontal Filter phase 0E

AD: Offset: 0xb37 | Byte Offset: 0x2cdc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd37 | Byte Offset: 0x34dc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf37 | Byte Offset: 0x3cdc | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P0EC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P0EC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P0EC3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P0EC2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P0EC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P0EC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.9.58 DC_<A/B/C>_WINC_<AD/BD/CD>_H_FILTER_HI_P0F_0

Window AD/BD/CD Horizontal Filter phase 0F

AD: Offset: 0xb38 | Byte Offset: 0x2ce0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd38 | Byte Offset: 0x34e0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf38 | Byte Offset: 0x3ce0 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:8	X	<AD/BD/CD>_H_FILTER_HI_P0FC5: MSB 5:4 of the lobe 5. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.
7:6	X	<AD/BD/CD>_H_FILTER_HI_P0FC4: MSB 7:6 of the lobe 4. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 4 are present in the B_H_FILTER_P** register.
5	X	<AD/BD/CD>_H_FILTER_HI_P0FC3: MSB 9:9 of the lobe 3.
4	X	<AD/BD/CD>_H_FILTER_HI_P0FC2: MSB 9:9 of the lobe 2.
3:2	X	<AD/BD/CD>_H_FILTER_HI_P0FC1: MSB 7:6 of the lobe 1. Signed coefficient. Bit 7 has the signed information. LSB 5:0 of the lobe 1 are present in the B_H_FILTER_P** register.
1:0	X	<AD/BD/CD>_H_FILTER_HI_P0FC0: MSB 5:4 of the lobe 0. Signed coefficient. Bit 5 has the signed information. LSB 3:0 are present in B_H_FILTER_P** register.

24.10 WIN_AD/BD/CD Registers

The registers under DC_WIN are double buffered.

24.10.1 DC_<A/B/C>_WIN_<AD/BD/CD>_WIN_OPTIONS_0

Window AD/BD/CD Options

Class: Display Window Settings

Display Window AD/BD/CD parameters

AD: Offset: 0xb80 | Byte Offset: 0x2e00 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxx00x0x0x0x0x0x0x0x0x0x0x0)
BD: Offset: 0xd80 | Byte Offset: 0x3600 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxx00x0x0x0x0x0x0x0x0x0x0x0)
CD: Offset: 0xf80 | Byte Offset: 0x3e00 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxx00x0x0x0x0x0x0x0x0x0x0x0)

Bit	Reset	Description
31	0x0	<AD/BD/CD>_H_FILTER_MODE: Horizontal filter mode. 0 = With previous Tegra horizontal coefficients widths. 0,5 - signed 5 bits 1,4 - signed 9 bits 2,3 - unsigned 9 bits 1 = New mode. With the expanded horizontal filter coefficients widths. 0,5- signed 5 bits 1,4- signed 7 bits 2,3- unsigned 9 bits 0 = OLD 1 = NEW
30	0x0	<AD/BD/CD>_WIN_ENABLE: Window AD/BD/CD Window enable 0 = DISABLE 1 = ENABLE
23	0x0	<AD/BD/CD>_INTERLACE_ENABLE: Window AD/BD/CD Interlace enable. This controls the fetch unit toggle between odd and even (normal) base addresses at the start of each field. Base address select between odd/even field is determined by HVTGEN field status. 0 = DISABLE 1 = ENABLE
22	0x0	<AD/BD/CD>_YUV_RANGE_EXPAND: Window AD/BD/CD Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0 = DISABLE 1 = ENABLE
20	0x0	<AD/BD/CD>_DV_ENABLE: Window AD/BD/CD Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	0x0	<AD/BD/CD>_CSC_ENABLE: Window AD/BD/CD Color Space Conversion Enable. This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes. 0 = DISABLE 1 = ENABLE
16	0x0	<AD/BD/CD>_CP_ENABLE: Window AD/BD/CD Color Palette Enable. This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
14	0x0	<AD/BD/CD>_V_FILTER_UV_ALIGN: Window AD/BD/CD V Filter UV Alignment. This is effective only when vertical scaling filter is enabled and only on these formats YCbCr420P, YUV420P, YCbCr422R, YUV422R, YCbCr422RA YUV422RA. When UV alignment is enabled, the chroma components are aligned to the even number of luma component lines. When disabled the chroma components are aligned to half a pixel below the corresponding even number of luma component lines. It is usually disabled unless the incoming video stream specifically indicates otherwise. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x0	<AD/BD/CD>_V_FILTER_ENABLE: Window AD/BD/CD V Filter Enable. This controls V scaling filter and is effective only for non-palletized color modes. If V filter is disabled, only one line is read from memory for each output line. NOTE: This feature is not supported,. This bit is ignored and is a don't care. 0 = DISABLE 1 = ENABLE
8	0x0	<AD/BD/CD>_H_FILTER_ENABLE: Window AD/BD/CD H Filter Enable. This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	0x0	<AD/BD/CD>_COLOR_EXPAND: Window AD/BD/CD 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
4	0x0	<AD/BD/CD>_SCAN_COLUMN: Window AD/BD/CD Scanning direction 0= Scan in horizontal direction (for 0 or 180 degrees) 1= Scan in vertical direction (for 90 or 270 degrees) 0 = DISABLE 1 = ENABLE
2	0x0	<AD/BD/CD>_V_DIRECTION: Window AD/BD/CD Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT
0	0x0	<AD/BD/CD>_H_DIRECTION: Window AD/BD/CD Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

24.10.2 DC_<A/B/C>_WIN_<AD/BD/CD>_BYTE_SWAP_0

Window AD/BD/CD Byte Swap

AD: Offset: 0xb81 │ Byte Offset: 0x2e04 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd81 │ Byte Offset: 0x3604 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf81 │ Byte Offset: 0x3e04 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	NOSWAP	<AD/BD/CD>_BYTE_SWAP: Window AD/BD/CD Byte Swap. This controls byte swap of frame data read from memory prior to any data processing in the display module. 00= no byte swap (3 2 1 0) 001= byte swap for each 2-byte word (2 3 0 1) 010= byte swap for each 4-byte word (0 1 2 3) 011= word swap for each 4-byte word (1 0 3 2) 100= byte0 swapped with byte2 (3 0 1 2) 101= left shift every byte (2 1 0 3) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW 4 = SWAP02 5 = SWAPLEFT

24.10.3 DC_<A/B/C>_WIN_<AD/BD/CD>_COLOR_DEPTH_0

Window AD/BD/CD Color Depth

This register describes the Window AD/BD/CD color surface format. Windows have different capabilities and support subsets of these formats.

Windows AD/BD/CD can use BYTE_SWAP to support additional formats.

Windows D/T do not support BYTE_SWAP. They support only the following formats:

- T_A8R8G8B8
- T_A8B8G8R8
- T_R5G6B5
- T_A4R4G4B4
- T_A1R5G5B5

For YCbCr data format, Cb and Cr are 8-bit unsigned values.

For YUV data format, U and V are 8-bit signed values.

YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P but the U and V are shared vertically.

YCbCr422RA is the same as YCbCr422R in memory, and YUV422RA is the same as YUV422R in memory. However, while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline.

Some formats have NVCF_ or T_ prefix aliases for NVIDIA surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word. The T_ names are preferred.

AD: Offset: 0xb83 │ Byte Offset: 0x2e0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)
BD: Offset: 0xd83 │ Byte Offset: 0x360c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)
CD: Offset: 0xf83 │ Byte Offset: 0x3e0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)

Bit	Reset	Description
6:0	B8G8R8A8	<AD/BD/CD>_COLOR_DEPTH: Window AD/BD/CD Color Depth. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422, TRUE_U8_Y8__V8_Y8 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P 22 = T_Y8___U8___V8_N422R 22 = YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE 23 = YUV422RP, YUV422R 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 52 = T_Y8___U8___V8_N444_TRUE, YUV444P 53 = T_Y8___U8V8_N420_TRUE, YVU420SP 54 = T_Y8___V8U8_N420_TRUE, YUV420SP 55 = T_Y8___U8V8_N422_TRUE, YVU422SP 56 = T_Y8___V8U8_N422_TRUE, YUV422SP 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP 59 = T_Y8___U8V8_N444_TRUE, YUV444SPYVU444SP 60 = T_Y8___V8U8_N444_TRUE, YVU444SPYUV444SP

24.10.4 DC_<A/B/C>_WIN_<AD/BD/CD>_POSITION_0

Window AD/BD/CD Position

This register defines the H position and size of Window AD/BD/CD after scaling (if there is any)

AD: Offset: 0xb84 │ Byte Offset: 0x2e10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd84 │ Byte Offset: 0x3610 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf84 │ Byte Offset: 0x3e10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	<AD/BD/CD>_V_POSITION: Window AD/BD/CD V Position. This is specified with respect to the top edge of active display area.
12:0	X	<AD/BD/CD>_H_POSITION: Window AD/BD/CD H Position. This is specified with respect to the left edge of active display area.

24.10.5 DC_<A/B/C>_WIN_<AD/BD/CD>_SIZE_0

Window AD/BD/CD Size

This register defines the V position and size of Window AD/BD/CD after scaling (if there is any)

Note: programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

AD: Offset: 0xb85 | Byte Offset: 0x2e14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd85 | Byte Offset: 0x3614 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf85 | Byte Offset: 0x3e14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	<AD/BD/CD>_V_SIZE: Window AD/BD/CD V Size (lines). This is the vertical size after scaling.
12:0	X	<AD/BD/CD>_H_SIZE: Window AD/BD/CD H Size (pixels). This is the horizontal size after scaling.

24.10.6 DC_<A/B/C>_WIN_<AD/BD/CD>_PRESCALED_SIZE_0

Window AD/BD/CD Pre-scaled Size

This register defines Window AD/BD/CD pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

AD: Offset: 0xb86 | Byte Offset: 0x2e18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd86 | Byte Offset: 0x3618 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf86 | Byte Offset: 0x3e18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	<AD/BD/CD>_V_PRESCALED_SIZE: Window AD/BD/CD V Pre-scaled Size (lines). In 420P/422R/422RA formats, it must be even.
14:0	X	<AD/BD/CD>_H_PRESCALED_SIZE: Window AD/BD/CD H Pre-scaled Size (bytes). In 420P and 422P formats, it must be even.

24.10.7 DC_<A/B/C>_WIN_<AD/BD/CD>_H_INITIAL_DDA_0

Window AD/BD/CD H Initial DDA

The first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though the user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly, with the V Initial DDA.

AD: Offset: 0xb87 | Byte Offset: 0x2e1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)
BD: Offset: 0xd87 | Byte Offset: 0x361c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)
CD: Offset: 0xf87 | Byte Offset: 0x3e1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	<AD/BD/CD>_H_INITIAL_DDA: Window AD/BD/CD H Initial DDA (4.12). This is typically programmed to 0.0

24.10.8 DC_<A/B/C>_WIN_<AD/BD/CD>_V_INITIAL_DDA_0

Window AD/BD/CD V Initial DDA

AD: Offset: 0xb88 | Byte Offset: 0x2e20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)
BD: Offset: 0xd88 | Byte Offset: 0x3620 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)
CD: Offset: 0xf88 | Byte Offset: 0x3e20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	<AD/BD/CD>_V_INITIAL_DDA: Window AD/BD/CD V Initial DDA (4.12). This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

24.10.9 DC_<A/B/C>_WIN_<AD/BD/CD>_DDA_INCREMENT_0

Window AD/BD/CD DDA Increment

DDA increment is typically calculated by dividing (Pre-scaled size in pixels - 1) by (Post-scaled size in pixels - 1). The result should be rounded up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered image this value can be slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 / (post_scaled_size_in_pixels - 1)) + initial_dda_bias, MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5) + initial_dda_bias, MAX)

Where the value of MAX is as follows:

For V_DDA_INCREMENT: 15.0 (0xF000)

For H_DDA_INCREMENT: 4.0 (0x4000) for 4 Bytes/pixel formats.

8.0 (0x8000) for 2 Bytes/pixel formats.

Where the value of initial_dda_bias is as follows:

For V_DDA_INCREMENT: initial_dda_bias = (v_intial_dda == negative) ? mod(v_initial_dda)/(post_scaled_size_in_pixels - 1) : 0 ;

For H_DDA_INCREMENT: initial_dda_bias = 0;

They are theoretically the biggest values that guarantee not displaying beyond an image boundary. If the DDA increment is less than 1.0, then the image is upscaled. If the DDA increment is more than 1.0, then the image is downscaled.

AD: Offset: 0xb89 | Byte Offset: 0x2e24 | Read/Write: R/W | Reset: 0x10001000 (0b00010000000000000001000000000000)
BD: Offset: 0xd89 | Byte Offset: 0x3624 | Read/Write: R/W | Reset: 0x10001000 (0b00010000000000000001000000000000)
CD: Offset: 0xf89 | Byte Offset: 0x3e24 | Read/Write: R/W | Reset: 0x10001000 (0b00010000000000000001000000000000)

Bit	Reset	Description
31:16	0x1000	<AD/BD/CD>_V_DDA_INCREMENT: Window AD/BD/CD Vertical DDA Increment (4.12). This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	0x1000	<AD/BD/CD>_H_DDA_INCREMENT: Window AD/BD/CD Horizontal DDA Increment (4.12). This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

24.10.10 DC_<A/B/C>_WIN_<AD/BD/CD>_LINE_STRIDE_0

Window AD/BD/CD Line Stride

AD: Offset: 0xb8a │ Byte Offset: 0x2e28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd8a │ Byte Offset: 0x3628 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf8a │ Byte Offset: 0x3e28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	<AD/BD/CD>_UV_LINE_STRIDE: Window AD/BD/CD Line Stride for Chroma. This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
15:0	X	<AD/BD/CD>_LINE_STRIDE: Window AD/BD/CD Line Stride. This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window A is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping) For tiled surface this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

24.10.11 DC_<A/B/C>_WIN_<AD/BD/CD>_DV_CONTROL_0

Window AD/BD/CD Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

AD: Offset: 0xb8e │ Byte Offset: 0x2e38 │ Read/Write: R/W │ Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xd8e │ Byte Offset: 0x3638 │ Read/Write: R/W │ Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xf8e │ Byte Offset: 0x3e38 │ Read/Write: R/W │ Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18:16	X	<AD/BD/CD>_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	<AD/BD/CD>_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	<AD/BD/CD>_DV_CONTROL_R: Digital Vibrance control for R

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending.

Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each window. If more than 1 windows color key is enabled, then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key ranges (Color Key 0 and Color Key 1) can be defined; they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap. If they do, the overlapped colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key parameters

For B4G4R4A4, B5G6R5A, and B5G6R5 modes, the color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, the color key is compared prior to the color palette, and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison. In all cases, the color key is compared prior to the horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

24.10.12 DC_<A/B/C>_WIN_<AD/BD/CD>_BLEND_LAYER_CONTROL_0

Window AD/BD/CD

AD: Offset: 0xb96 | Byte Offset: 0x2e58 | Read/Write: R/W | Reset: 0x01000000 (0bxxxx00010000000000000000000000)
BD: Offset: 0xd96 | Byte Offset: 0x3658 | Read/Write: R/W | Reset: 0x01000000 (0bxxxx00010000000000000000000000)
CD: Offset: 0xf96 | Byte Offset: 0x3e58 | Read/Write: R/W | Reset: 0x01000000 (0bxxxx00010000000000000000000000)

Bit	Reset	Description
27:25	0x0	<AD/BD/CD>_COLOR_KEY_SELECT: 0 = NONE 1 = WINDOWA_KEY0 2 = WINDOWA_KEY1 3 = WINDOWB_KEY0 4 = WINDOWB_KEY1 5 = WINDOWC_KEY0 6 = WINDOWC_KEY1
24	BLEND_BYPASS	<AD/BD/CD>_BLEND_BYPASS: 0 = BLEND_ENABLE 1 = BLEND_BYPASS
23:16	0x0	<AD/BD/CD>_K2
15:8	0x0	<AD/BD/CD>_K1
7:0	0x0	<AD/BD/CD>_WINDOW_LAYER_DEPTH

24.10.13 DC_<A/B/C>_WIN_<AD/BD/CD>_BLEND_MATCH_SELECT_0

AD: Offset: 0xb97 | Byte Offset: 0x2e5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)
BD: Offset: 0xd97 | Byte Offset: 0x365c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)
CD: Offset: 0xf97 | Byte Offset: 0x3e5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

> ***Note:*** *Refer to Section 24.1.4.2: Blending for algorithms/equations used.*

Bit	Reset	Description
13:12	0x0	<AD/BD/CD>_BLEND_FACTOR_DST_ALPHA_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	<AD/BD/CD>_BLEND_FACTOR_SRC_ALPHA_MATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	<AD/BD/CD>_BLEND_FACTOR_DST_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	<AD/BD/CD>_BLEND_FACTOR_SRC_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

24.10.14 DC_<A/B/C>_WIN_<AD/BD/CD>_BLEND_NOMATCH_SELECT_0

AD: Offset: 0xb98 | Byte Offset: 0x2e60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)
BD: Offset: 0xd98 | Byte Offset: 0x3660 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)
CD: Offset: 0xf98 | Byte Offset: 0x3e60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Note: *Refer to Section 24.1.4.2: Blending for algorithms/equations used.*

Bit	Reset	Description
13:12	0x0	<AD/BD/CD>_BLEND_FACTOR_DST_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	<AD/BD/CD>_BLEND_FACTOR_SRC_ALPHA_NOMATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	<AD/BD/CD>_BLEND_FACTOR_DST_COLOR_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1
2:0	0x0	<AD/BD/CD>_BLEND_FACTOR_SRC_COLOR_NOMATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

24.10.15 DC_<A/B/C>_WIN_<AD/BD/CD>_BLEND_ALPHA_1BIT_0

AD: Offset: 0xb99 | Byte Offset: 0x2e64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)
BD: Offset: 0xd99 | Byte Offset: 0x3664 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)
CD: Offset: 0xf99 | Byte Offset: 0x3e64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	<AD/BD/CD>_BLEND_WEIGHT1: This is used only for 1-bit alpha and alpha value of 1.
7:0	0x0	<AD/BD/CD>_BLEND_WEIGHT0: This is used only for 1-bit alpha and alpha value of 0.

24.11 WINBUF_AD/BD/CD Registers

Note: *In the register and field names, <A/B/C> indicates A, B, or C, and <AD/BD/CD> indicates AD, BD, or CD, depending on the window selection.*

24.11.1 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_0

Window AD/BD/CD Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally, START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see ”For linear address mode” in Programming Restrictions below.)

Note that “beginning of the window” here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, starting address of a window is calculated as follows by hardware:
  - non-YUV-planar modes:

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes:

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.
- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-YUV-planar mode:

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

i. For tiled address mode:

Image surface can only be aligned to multiples of 256, thus the following restrictions.

- START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
- BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
- LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
- ADDR_H_OFFSET needs to be even in YUV planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset.
- ADDR_V_OFFSET needs to be a multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats.

ii. For linear address mode:

Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)

- For all formats:
  - START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.
- For 16-bpp formats,
  - (START_ADDR+H_OFFSET) need to be a multiple of 2.
- For 32-bpp formats,
  - (START_ADDR+H_OFFSET) needs to be a multiple of 4.
- For YUV planar formats:
  - BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE is a multiple of 8, UV_BUF_STRIDE is a multiple of 4.
  - LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.
  - ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset. .
  - ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset. .

iii. Memory allocation for non-host triggered case:

When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame. Also buffer wraparound must not occur in the middle of the displayed part of the frame.

The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

AD: Offset: 0xbc0 | Byte Offset: 0x2f00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdc0 | Byte Offset: 0x3700 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfc0 | Byte Offset: 0x3f00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_START_ADDR: Window AD/BD/CD Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

24.11.2 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_U_0

Window AD/BD/CD Start Address for U plane

AD: Offset: 0xbc2 | Byte Offset: 0x2f08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdc2 | Byte Offset: 0x3708 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfc2 | Byte Offset: 0x3f08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_START_ADDR_U: Window AD/BD/CD Start Address for U plane. This is a byte address. Note: You should write both U and V plane addresses to the same value for semi-planar to work correctly.

24.11.3 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_V_0

Window AD/BD/CD Start Address for V plane

AD: Offset: 0xbc4 | Byte Offset: 0x2f10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdc4 | Byte Offset: 0x3710 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfc4 | Byte Offset: 0x3f10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_START_ADDR_V: Window AD/BD/CD Start Address for V plane. This is a byte address. Note: You should write both U and V plane addresses to the same value for semi-planar to work correctly.

24.11.4 DC_<A/B/C>_WINBUF_<AD/BD/CD>_ADDR_H_OFFSET_0

Window AD/BD/CD Horizontal Address Offset

AD: Offset: 0xbc6 | Byte Offset: 0x2f18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdc6 | Byte Offset: 0x3718 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfc6 | Byte Offset: 0x3f18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_ADDR_H_OFFSET: Window AD/BD/CD Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

24.11.5 DC_<A/B/C>_WINBUF_<AD/BD/CD>_ADDR_V_OFFSET_0

Window AD/BD/CD Vertical Address Offset

AD: Offset: 0xbc8 | Byte Offset: 0x2f20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdc8 | Byte Offset: 0x3720 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfc8 | Byte Offset: 0x3f20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_ADDR_V_OFFSET: Window AD/BD/CD Vertical address offset. This is a line number. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies vertical offset of Y plane. The vertical offsets of U/V plane is derived by hardware.

24.11.6 DC_<A/B/C>_WINBUF_<AD/BD/CD>_UFLOW_STATUS_0

Window AD/BD/CD FIFO Underflow Status Register

AD: Offset: 0xbca | Byte Offset: 0x2f28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdca | Byte Offset: 0x3728 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfca | Byte Offset: 0x3f28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

24.11.7 DC_<A/B/C>_WINBUF_<AD/BD/CD>_SURFACE_KIND_0

Window AD/BD/CD Surface Kind

AD: Offset: 0xbcb | Byte Offset: 0x2f2c | Read/Write: R/W | Reset: 0x00000010 (0b0xxxxxxxxxxxxxxxxxxxxxxxx001xx00)
BD: Offset: 0xdcb | Byte Offset: 0x372c | Read/Write: R/W | Reset: 0x00000010 (0b0xxxxxxxxxxxxxxxxxxxxxxxx001xx00)
CD: Offset: 0xfcb | Byte Offset: 0x3f2c | Read/Write: R/W | Reset: 0x00000010 (0b0xxxxxxxxxxxxxxxxxxxxxxxx001xx00)

Bit	Reset	Description
31	0x0	<AD/BD/CD>_BLX4_CYA: Force BLx4 fetches if surface is Block Linear. 0 = DISABLE 1 = ENABLE
6:4	0x1	<AD/BD/CD>_BLOCK_HEIGHT: Block Height: For block linear, height of block in GOBs 0 = HEIGHT_1 1 = HEIGHT_2 2 = HEIGHT_4 3 = HEIGHT_8 4 = HEIGHT_16 5 = HEIGHT_32
1:0	0x0	<AD/BD/CD>_SURFACE_KIND: Surface Kind: Pitched, Tiled, or Block Linear 0 = PITCH 1 = TILED 2 = BL_16B2

24.11.8 DC_<A/B/C>_WINBUF_<AD/BD/CD>_SURFACE_WEIGHT_0

Window AD/BD/CD Surface Weights

AD: Offset: 0xbcc | Byte Offset: 0x2f30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)
BD: Offset: 0xdcc | Byte Offset: 0x3730 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)
CD: Offset: 0xfcc | Byte Offset: 0x3f30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
6:5	X	<AD/BD/CD>_SURFACE_WEIGHT_V: Window AD/BD/CD V Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
4:3	X	<AD/BD/CD>_SURFACE_WEIGHT_U: Window AD/BD/CD U or UV Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
2:1	X	<AD/BD/CD>_SURFACE_WEIGHT_Y: Window AD/BD/CD Y or packed Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
0	0x0	<AD/BD/CD>_SURFACE_WEIGHT_OVERRIDE: Weight Override 0 = DISABLE 1 = ENABLE

24.11.9 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_HI_0

Window AD/BD/CD Higher 2 bits of Start Address

Makes the start address 34 bits by adding the higher 2 bits below.

AD: Offset: 0xbcd | Byte Offset: 0x2f34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
BD: Offset: 0xdcd | Byte Offset: 0x3734 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
CD: Offset: 0xfcd | Byte Offset: 0x3f34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	<AD/BD/CD>_START_ADDR_HI: Window AD/BD/CD Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

24.11.10 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_HI_U_0

Window AD/BD/CD Higher 2 bits of Start Address for U Plane

Makes the start address 34 bits by adding the higher 2 bits below.

AD: Offset: 0xbcf | Byte Offset: 0x2f3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
BD: Offset: 0xdcf | Byte Offset: 0x373c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
CD: Offset: 0xfcf | Byte Offset: 0x3f3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	<AD/BD/CD>_START_ADDR_HI_U: Window AD/BD/CD Start Address for U plane. This is a byte address.

24.11.11 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_HI_V_0

Window AD/BD/CD Higher 2 bits of Start Address for V Plane

Makes the start address 34 bits by adding the higher 2 bits below.

AD: Offset: 0xbd1 | Byte Offset: 0x2f44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
BD: Offset: 0xdd1 | Byte Offset: 0x3744 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
CD: Offset: 0xfd1 | Byte Offset: 0x3f44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	<AD/BD/CD>_START_ADDR_HI_V: Window AD/BD/CD Higher 2 bits of Start Address for V plane. This is a byte address.

24.11.12 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_FIELD2_0

Window AD/BD/CD Start Address

Interlace/Stereo support. 64-bit base address for the odd field/right frame.

AD: Offset: 0xbd3 | Byte Offset: 0x2f4c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdd3 | Byte Offset: 0x374c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfd3 | Byte Offset: 0x3f4c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_START_ADDR_FIELD2: Window AD/BD/CD Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

24.11.13 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_FIELD2_U_0

Window AD/BD/CD Start Address for U plane

AD: Offset: 0xbd5 | Byte Offset: 0x2f54 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdd5 | Byte Offset: 0x3754 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfd5 | Byte Offset: 0x3f54 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_START_ADDR_FIELD2_U: Window AD/BD/CD Start Address for U plane. This is a byte address.

24.11.14 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_FIELD2_V_0

Window AD/BD/CD Start Address for V plane

AD: Offset: 0xbd7 | Byte Offset: 0x2f5c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdd7 | Byte Offset: 0x375c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfd7 | Byte Offset: 0x3f5c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_START_ADDR_FIELD2_V: Window AD/BD/CD Start Address for V plane. This is a byte address.

24.11.15 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_FIELD2_HI_0

Window AD/BD/CD Higher 2 bits of Start Address

Makes the start address 34 bits by adding the higher 2 bits below.

AD: Offset: 0xbd9 | Byte Offset: 0x2f64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
BD: Offset: 0xdd9 | Byte Offset: 0x3764 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
CD: Offset: 0xfd9 | Byte Offset: 0x3f64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	<AD/BD/CD>_START_ADDR_FIELD2_HI: Window AD/BD/CD Start Address. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the start address for the Y plane.

24.11.16 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_FIELD2_HI_U_0

Window AD/BD/CD Higher 2 bits of Start Address for U plane

Makes the start address 34 bits by adding the higher 2 bits below.

AD: Offset: 0xbdb | Byte Offset: 0x2f6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
BD: Offset: 0xddb | Byte Offset: 0x376c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
CD: Offset: 0xfdb | Byte Offset: 0x3f6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	<AD/BD/CD>_START_ADDR_FIELD2_HI_U: Window AD/BD/CD Start Address for U plane. This is a byte address.

24.11.17 DC_<A/B/C>_WINBUF_<AD/BD/CD>_START_ADDR_FIELD2_HI_V_0

Window AD/BD/CD Higher 2 bits of Start Address for V plane

Makes the start address 34 bits by adding the higher 2 bits below.

AD: Offset: 0xbdd | Byte Offset: 0x2f74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
BD: Offset: 0xddd | Byte Offset: 0x3774 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
CD: Offset: 0xfdd | Byte Offset: 0x3f74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	<AD/BD/CD>_START_ADDR_FIELD2_HI_V: Window AD/BD/CD Higher 2 bits of Start Address for V plane. This is a byte address.

24.11.18 DC_<A/B/C>_WINBUF_<AD/BD/CD>_ADDR_H_OFFSET_FIELD2_0

Window AD/BD/CD Horizontal address offset

AD: Offset: 0xbdf | Byte Offset: 0x2f7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xddf | Byte Offset: 0x377c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfdf | Byte Offset: 0x3f7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_ADDR_H_OFFSET_FIELD2: Window AD/BD/CD Horizontal address offset. This is a byte address. The LSB is not used for the 16-bpp non-planar pixel format, and the last 2 LSBs are not used for the 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies the horizontal offset of the Y plane. The horizontal offsets of U/V plane are derived by hardware.

24.11.19 DC_<A/B/C>_WINBUF_<AD/BD/CD>_ADDR_V_OFFSET_FIELD2_0

Window AD/BD/CD Vertical address offset

AD: Offset: 0xbe1 | Byte Offset: 0x2f84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xde1 | Byte Offset: 0x3784 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfe1 | Byte Offset: 0x3f84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_ADDR_V_OFFSET_FIELD2: Window AD/BD/CD Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of the Y plane. The vertical coordinate of U/V plane is derived by hardware.

24.11.20 DC_<A/B/C>_WINBUF_<AD/BD/CD>_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

AD: Offset: 0xbe4 | Byte Offset: 0x2f90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)
BD: Offset: 0xde4 | Byte Offset: 0x3790 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)
CD: Offset: 0xfe4 | Byte Offset: 0x3f90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	<AD/BD/CD>_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

24.11.21 DC_<A/B/C>_WINBUF_<AD/BD/CD>_UFLOW_DBG_PIXEL_0

AD: Offset: 0xbe5 | Byte Offset: 0x2f94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xde5 | Byte Offset: 0x3794 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xfe5 | Byte Offset: 0x3f94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_UFLOW_DBG_PIXEL

24.11.22 DC_<A/B/C>_WINBUF_<AD/BD/CD>_UFLOW_THRESHOLD_0

AD: Offset: 0xbe6 | Byte Offset: 0x2f98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)
BD: Offset: 0xde6 | Byte Offset: 0x3798 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)
CD: Offset: 0xfe6 | Byte Offset: 0x3f98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	<AD/BD/CD>_UFLOW_THRESHOLD

24.11.23 DC_<A/B/C>_WINBUF_<AD/BD/CD>_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra X1 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the Vblank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for MC, it has a high potential to starve out other clients.

SPOOL_UP_CTRL = PROGRAMMABLE, SPOOL_UP_DURATION = 1 - This corresponds to the Tegra 3 behavior. Since the fetch starts only 1 line prior to the active scan line of a window, this has high probability of causing underflow, but has a lesser impact on the other MC clients.

So, this register programmability programs a method to modulate the spool up time between the best and worst, if required, when other MC clients are present. For example when GPU/VIC is active, we may not want a very aggressive spool up and program some definite spool up duration. So, spool up time is a general knob, which would be a function of the resolution, bpp, active clients, memory speed, etc.

SPOOL_UP_EDGE:- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- In this case if SPOOL_UP_EDGE is programmed to NEGEDGE. This ensures that if any act_req is sent during the time vcounter = 0(at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value. The downside of programming it to NEGDGE is that we lose out on 1 line clock worth of spool up. This is required based upon Tegra compatibility and the way tests are written.
- In this case if SPOOL_UP_EDGE is programmed to POSEDGE. Then the memfetch would start fetching at the very beginning of the frame start pulse (1 line clock wide) and would give maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above

FOR SPOOL_UP_CTRL = PROGRAMMABLE:

- Special care must be taken here. If SPOOL_UP_DURATION is set to 1. SPOOL_UP_EDGE must be set to POSEDGE to allow for at least 1 line worth of spool up.
- If SPOOL_UP_DURATION is > 1. Then SPOOL_UP_EDGE can be either POSEDGE or NEGEDGE. If it is NEGEDGE, then the effective spool up duration would be 1 line clock less than the SPOOL_UP_DURATION.

AD: Offset: 0xbe7 | Byte Offset: 0x2f9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
BD: Offset: 0xde7 | Byte Offset: 0x379c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
CD: Offset: 0xfe7 | Byte Offset: 0x3f9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	<AD/BD/CD>_SPOOL_UP_DURATION
1	0x0	<AD/BD/CD>_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE
0	0x0	<AD/BD/CD>_SPOOL_UP_CTRL: 0 = MAX 1 = PROGRAMMABLE

24.11.24 DC_<A/B/C>_WINBUF_<AD/BD/CD>_SCALEFACTOR_THRESHOLD_0

Registers for latency allowance scaling. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above HWM or below LWM.

AD: Offset: 0xbe8 | Byte Offset: 0x2fa0 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)
BD: Offset: 0xde8 | Byte Offset: 0x37a0 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)
CD: Offset: 0xfe8 | Byte Offset: 0x3fa0 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	<AD/BD/CD>_SF_LWM_THRESHOLD
15:0	0xffff	<AD/BD/CD>_SF_HWM_THRESHOLD

24.11.25 DC_<A/B/C>_WINBUF_<AD/BD/CD>_LATENCY_THRESHOLD_0

The LATENCY_THRESHOLD register controls the behavior of the rdy4_latency_event from the display to the MC.

If B_RDY4LATENCY_THRESHOLD_ENABLE is set to DISABLE, the sideband rdy4_latency_event is always asserted, indicating that the display is always ready for a DVFS event.

The B_RDY4LATENCY_THRESHOLD field indicates the required occupancy of line buffers of windows below which the sideband rdy4_latency_event is deasserted. The occupancy figure is in units of 64B atoms for memfetch windows and in units of scan lines for regular windows.

The B_RDY4LATENCY_SPOOLUP_DURATION field indicates the number of scan lines during when the sideband will remain asserted (DVFS is allowed). This takes affect only when B_RDY4LATENCY_SPOOLUP_CTRL is set to ALLOW.

If the B_RDY4LATENCY_SPOOLUP_CTRL field is set to DISALLOW, the above register field is not affected and DVFS is disallowed until the threshold is reached.

AD: Offset: 0xbe9 │ Byte Offset: 0x2fa4 │ Read/Write: R/W │ Reset: 0x0000ffff (0b00x00000000000001111111111111111)
BD: Offset: 0xde9 │ Byte Offset: 0x37a4 │ Read/Write: R/W │ Reset: 0x0000ffff (0b00x00000000000001111111111111111)
CD: Offset: 0xfe9 │ Byte Offset: 0x3fa4 │ Read/Write: R/W │ Reset: 0x0000ffff (0b00x00000000000001111111111111111)

Bit	Reset	Description
31	DISABLE	<AD/BD/CD>_RDY4LATENCY_THRESHOLD_ENABLE: 0 = DISABLE 1 = ENABLE
30	DISALLOW	<AD/BD/CD>_RDY4LATENCY_SPOOLUP_CTRL: 0 = DISALLOW 1 = ALLOW
28:16	0x0	<AD/BD/CD>_RDY4LATENCY_SPOOLUP_DURATION
15:0	0xffff	<AD/BD/CD>_RDY4LATENCY_THRESHOLD

24.11.26 DC_<A/B/C>_WINBUF_<AD/BD/CD>_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

AD: Offset: 0xbea │ Byte Offset: 0x2fa8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)
BD: Offset: 0xdea │ Byte Offset: 0x37a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)
CD: Offset: 0xfea │ Byte Offset: 0x3fa8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	0x0	<AD/BD/CD>_DEBUG_ENABLE:reg to enable FGCG for flops added to debug signals in memfetch.
15	0x0	<AD/BD/CD>_ALIGN_FIFO_NON_IDLE: align FIFO idle at end of frame. 0= align FIFO is idle at end of frame. 1= align FIFO is not idle at end of frame
14	0x0	<AD/BD/CD>_PIPE1_FIFO_NON_IDLE: pipe1 FIFO idle at end of frame. 0= pipe1 FIFO is idle at end of frame. 1= pipe1 FIFO is not idle at end of frame
13	0x0	<AD/BD/CD>_PIPE0_FIFO_NON_IDLE: pipe0 FIFO idle at end of frame. 0= pipe0 FIFO is idle at end of frame. 1= pipe0 FIFO is not idle at end of frame

Bit	Reset	Description
12	0x0	<AD/BD/CD>_FRAME_FIFO_NON_IDLE: frame FIFO idle at end of frame. 0= frame FIFO is idle at end of frame. 1= frame FIFO is not idle at end of frame
11	0x0	<AD/BD/CD>_SURFACE_FIFO_NON_IDLE: surface FIFO idle at end of frame. 0= surface FIFO is idle at end of frame. 1= surface FIFO is not idle at end of frame
10	0x0	<AD/BD/CD>_TAG_FIFO_NON_IDLE: tag FIFO idle at end of frame. 0= tag FIFO is idle at end of frame. 1= tag FIFO is not idle at end of frame
9	0x0	<AD/BD/CD>_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	<AD/BD/CD>_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
7	0x0	<AD/BD/CD>_DP_VALID_NON_IDLE: DP lines valid at end of frame. 0= DP lines are not valid at end of frame. 1= DP lines are valid at end of frame
6	0x0	<AD/BD/CD>_V_SSM_NON_IDLE: V SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
5	0x0	<AD/BD/CD>_U_SSM_NON_IDLE: U SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
4	0x0	<AD/BD/CD>_Y_SSM_NON_IDLE: Y SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
3	0x0	<AD/BD/CD>_DP_POOL_AVAIL: DP thinks there is data in the pool at end of frame. 0= No pool data available at end of frame. 1= pool data available at end of frame
2	0x0	<AD/BD/CD>_POOL_NOT_EMPTY: Status of pool at end of frame. 0= pool is empty at end of frame. 1= pool is not empty at end of frame
1	0x0	<AD/BD/CD>_UNDERFLOW_LINE1: Underflow of line0 0= No underflow 1= line0 underflowed
0	0x0	<AD/BD/CD>_UNDERFLOW_LINE0: Underflow of line0 0= No underflow 1= line0 underflowed

24.11.27 DC_<A/B/C>_WINBUF_<AD/BD/CD>_MEMFETCH_CONTROL_0

Note: *MEMFETCH_CLK_GATE_OVERRIDE should only be enabled AFTER the vertical scaler settings in the active copy - VDDA_INCREMENT is properly set up.*

Memfetch Control Register

AD: Offset: 0xbeb │ Byte Offset: 0x2fac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001
BD: Offset: 0xdeb │ Byte Offset: 0x37ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001
CD: Offset: 0xfeb │ Byte Offset: 0x3fac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001

Bit	Reset	SW Default	Description
1	0x0	NONE	<AD/BD/CD>_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	ENABLE	<AD/BD/CD>_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

24.11.28 DC_<A/B/C>_WINBUF_<AD/BD/CD>_OCCUPANCY_THROTTLE_0

AD: Offset: 0xbec │ Byte Offset: 0x2fb0 │ Read/Write: R/W │ Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)
BD: Offset: 0xdec │ Byte Offset: 0x37b0 │ Read/Write: R/W │ Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)
CD: Offset: 0xfec │ Byte Offset: 0x3fb0 │ Read/Write: R/W │ Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	<AD/BD/CD>_OCCUPANCY_MAX_THRESHOLD
0	0x0	<AD/BD/CD>_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

24.11.29 DC_<A/B/C>_WINBUF_<AD/BD/CD>_SCRATCH_REGISTER_0_0

AD: Offset: 0xbed │ Byte Offset: 0x2fb4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)
BD: Offset: 0xded │ Byte Offset: 0x37b4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)
CD: Offset: 0xfed │ Byte Offset: 0x3fb4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	<AD/BD/CD>_SCRATCH_REGISTER_0:Scratch register 0

24.11.30 DC_<A/B/C>_WINBUF_<AD/BD/CD>_SCRATCH_REGISTER_1_0

AD: Offset: 0xbee │ Byte Offset: 0x2fb8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)
BD: Offset: 0xdee │ Byte Offset: 0x37b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)
CD: Offset: 0xfee │ Byte Offset: 0x3fb8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	<AD/BD/CD>_SCRATCH_REGISTER_1:Scratch register 1

24.12 WINBUF_AD/BD/CD CDE Registers

Note: *In the register and field names, <A/B/C> indicates A, B, or C, and <AD/BD/CD> indicates AD, BD, or CD, depending on the window selection.*

24.12.1 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_CONTROL_0

These WINBUF A CDE registers control window (WIN) parameters.

Note that there are three copies of these registers for windows A, B, and C. Only comptag base address registers are triple-buffered to be consistent with display surface start address buffering (this is required to support immediate flips).All other registers are double-buffered so that flips can be atomic and consistent. Registers will be activated by the WIN_x_ACT_REQ/UPDATE in display.

This register implements per surface compression enables, drawing direction, scanning direction, surface traversal control, and diagnostics to reset Request and Response side on Frame Start.

AD: Offset: 0xbef | Byte Offset: 0x2fbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxxxxxxxx0)
BD: Offset: 0xdef | Byte Offset: 0x37bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxxxxxxxx0)
CD: Offset: 0xfef | Byte Offset: 0x3fbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxxxxxxxx0)

Bit	Reset	Description
13	0x0	<AD/BD/CD>_CLEARRESPONSEONFRAMESTART. The CLEARRESPONSEONFRAMESTART bit is a diagnostic bit to reset response block state machines and buffers with FrameStart: 0 = DISABLE 1 = ENABLE
12	0x0	<AD/BD/CD>_CLEARREQUESTONFRAMESTART. The CLEARREQUESTONFRAMESTART bit is a diagnostic bit to reset request block state machines and buffers with FrameStart : 0 = DISABLE 1 = ENABLE
0	0x0	<AD/BD/CD>_ENABLESURFACE0. The ENABLE_SURFACE0 bit identifies if compression is enabled for surface 0.For Display, only surface 0 is implemented.: 0 = DISABLE 1 = ENABLE

24.12.2 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_COMPTAG_BASE_0_0

This register implements per surface base address of compression bit storage.

AD: Offset: 0xbf0 | Byte Offset: 0x2fc0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)
BD: Offset: 0xdf0 | Byte Offset: 0x37c0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)
CD: Offset: 0xff0 | Byte Offset: 0x3fc0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	<AD/BD/CD>_BASEADDRESS. The BASEADDRESS bits identify compression bit storage Base Address for surface N. For Display, only surface 0 is implemented

24.12.3 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_COMPTAG_BASEHI_0_0

This register implements per surface base address high of compression bit storage.

AD: Offset: 0xbf2 | Byte Offset: 0x2fc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)
BD: Offset: 0xdf2 | Byte Offset: 0x37c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)
CD: Offset: 0xff2 | Byte Offset: 0x3fc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	<AD/BD/CD>_BASEADDRESSHI. The BASEADDRESSHI bits identify compression bit storage Base Address Hi for surface N. For Display, only surface 0 is implemented.

24.12.4 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_ZBC_COLOR_0_0

This register implements one ZBC Color per surface.

AD: Offset: 0xbf4 | Byte Offset: 0x2fd0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)
BD: Offset: 0xdf4 | Byte Offset: 0x37d0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)
CD: Offset: 0xff4 | Byte Offset: 0x3fd0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	<AD/BD/CD>_PIXELCOLOR. The PIXELCOLOR bits identify the ZBC Color for surface N.

24.12.5 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_SURFACE_OFFSET_0_0

This register implements X and Y offset of surface N.

AD: Offset: 0xbf5 | Byte Offset: 0x2fd4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000xxxx000000000000)
BD: Offset: 0xdf5 | Byte Offset: 0x37d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000xxxx000000000000)
CD: Offset: 0xff5 | Byte Offset: 0x3fd4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000xxxx000000000000)

Bit	Reset	Description
27:16	0x0	<AD/BD/CD>_YOFFSET. The YOFFSET bits identify the Y offset for surface N
11:0	0x0	<AD/BD/CD>_XOFFSET. The XOFFSET bits identify the X offset for surface N.

24.12.6 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_CTB_ENTRY_0_0

This register implements per surface Maximum CTB entry count (in terms of CTB entry pairs) for Surfaces 0-3. For Display only, surface 0 is implemented.

AD: Offset: 0xbf6 | Byte Offset: 0x2fd8 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000010)
BD: Offset: 0xdf6 | Byte Offset: 0x37d8 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000010)
CD: Offset: 0xff6 | Byte Offset: 0x3fd8 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000010)

Bit	Reset	Description
7:0	0x2	<AD/BD/CD>_MAXENTRYCOUNT0. The MAXENTRYCOUNT{i} bits identify the per surface Maximum CTB entry count(in terms of CTB entry pairs) for Surfaces 0-3.For Display only surface 0 is implemented.

24.12.7 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_CG_SW_OVR_0

This register implements SW override to enable/disable CDE clock-gating.

AD: Offset: 0xbf7 | Byte Offset: 0x2fdc | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)
BD: Offset: 0xdf7 | Byte Offset: 0x37dc | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)
CD: Offset: 0xff7 | Byte Offset: 0x3fdc | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	<AD/BD/CD>_CG. The CG bit identifies the CG software override to enable/disable CDE clock gating: 0 = DISABLE 1 = ENABLE

24.12.8 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_PM_CONTROL_0

This register is a select for PM counters implemented in the CDE.

AD: Offset: 0xbf8 | Byte Offset: 0x2fe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)
BD: Offset: 0xdf8 | Byte Offset: 0x37e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)
CD: Offset: 0xff8 | Byte Offset: 0x3fe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	<AD/BD/CD>_SELECT. The SELECT bits identify the select lines for PM counters implemented in the CDE: 0 = CTB_LOOKUP_HIT 1 = CTB_LOOKUP_MISS 2 = UNCOMP_1TO1 3 = ARITHMETIC_2TO1 4 = REDUCTION_8TO1 5 = ZBC 6 = CSR_FETCH_64B 7 = CTB_FETCH_64B 8 = PIXEL_FETCH_64B 9 = PIXEL_FETCH_32B 10 = CTB_READS_SAMPLED 11 = CTB_READ_RSP_LATENCY 12 = PIXEL_READS_SAMPLED 13 = PIXEL_READ_RSP_LATENCY

24.12.9 DC_<A/B/C>_WINBUF_<AD/BD/CD>_CDE_PM_COUNTER_0

This register is one of the read-back PM counters implemented in the CDE. Selected based on CDE_PM_CONTROL_SELECT.

AD: Offset: 0xbf9 | Byte Offset: 0x2fe4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
BD: Offset: 0xdf9 | Byte Offset: 0x37e4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
CD: Offset: 0xff9 | Byte Offset: 0x3fe4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<AD/BD/CD>_VALUE. The VALUE bits identify the value of PM counter based on select lines.

24.13 Window DD/TD (WIN_DD/TD) Registers

These registers control window DD and window TD parameters.

The registers under DC_WIN are double buffered.

Window TD: only 2BPP and 4BPP pitched linear format. None of the features listed in Windows A/B/C. The registers under DC_WIN_T are only accessed directly and not triple buffered.

> ***Note:*** *In the register and field names, <D/T> indicates D or T, and <DD/TD> indicates DD or TD, depending on the window selection.*

24.13.1 DC_<D/T>_WIN_<DD/TD>_WIN_OPTIONS_0

Window DD/TD Options

Class: Display Window Settings

Display Window DD/TD parameters

DD: Offset: 0x80 | Byte Offset: 0x200 | Read/Write: R/W | Reset: 0x00000000 (0bx0xxxxxxxxxxxxxxxxxxxxxxx0xxxxxx)
TD: Offset: 0x100 | Byte Offset: 0x400 | Read/Write: R/W | Reset: 0x00000000 (0bx0xxxxxxxxxxxxxxxxxxxxxxx0xxxxxx)

Bit	Reset	Description
30	0x0	<DD/TD>_WIN_ENABLE: Window DD/TD Window enable 0 = DISABLE 1 = ENABLE
6	0x0	<DD/TD>_COLOR_EXPAND: Window DD/TD 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled, the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE

24.13.2 DC_<D/T>_WIN_<DD/TD>_COLOR_DEPTH_0

Window DD/TD Color Depth

This register describes the Window DD/TD color surface format. Windows have different capabilities and support subsets of these formats.

Windows A/B/C can use BYTE_SWAP to support additional formats.

Windows D/T do not support BYTE_SWAP. They support only the following formats:

- T_A8R8G8B8
- T_A8B8G8R8
- T_R5G6B5

- T_A4R4G4B4
- T_A1R5G5B5

For the YCbCr data format, Cb and Cr are 8-bit unsigned values. For the YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P, but the U and V are shared vertically. YCbCr422RA is the same as YCbCr422R in memory and YUV422RA is the same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline.

Some formats have NVCF_ or T_ prefix aliases for Nvidia surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

The T_ names are preferred.

DD: Offset: 0x83 | Byte Offset: 0x20c | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)
TD: Offset: 0x103 | Byte Offset: 0x40c | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0001100)

Bit	Reset	Description
6:0	B8G8R8A8	<DD/TD>_COLOR_DEPTH: Window D Color Depth. T_A8R8G8B8 - 12 T_A8B8G8R8 - 13 T_R5G6B5 - 6 T_A4R4G4B4 - 4 T_A1R5G5B5 - 5

24.13.3 DC_<D/T>_WIN_<DD/TD>_POSITION_0

Window DD/TD Position

This register defines the H position and size of Window DD/TD after scaling (if there is any).

DD: Offset: 0x84 | Byte Offset: 0x210 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x104 | Byte Offset: 0x410 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	<DD/TD>_V_POSITION: Window DD/TD V Position. This is specified with respect to the top edge of active display area.
12:0	X	<DD/TD>_H_POSITION: Window DD/TD H Position. This is specified with respect to the left edge of active display area.

24.13.4 DC_<D/T>_WIN_<DD/TD>_SIZE_0

Window DD/TD Size

This register defines the V position and size of Window DD/TD after scaling (if there is any).

> ***Note:*** *Programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.*

DD: Offset: 0x85 | Byte Offset: 0x214 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x105 | Byte Offset: 0x414 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	<DD/TD>_V_SIZE: Window DD/TD V Size (lines). This is the vertical size after scaling.

Bit	Reset	Description
12:0	X	<DD/TD>_H_SIZE: Window DD/TD H Size (pixels). This is the horizontal size after scaling.

24.13.5 DC_<D/T>_WIN_<DD/TD>_LINE_STRIDE_0

Window DD/TD Line Stride

DD: Offset: 0x8a | Byte Offset: 0x228 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x10a | Byte Offset: 0x428 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	<DD/TD>_LINE_STRIDE: Window DD/TD Line Stride. This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window DD/TD is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be in multiples of 8 (16 if tiled or in horizontal flipping) For tiled surface this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

24.13.6 DC_<D/T>_WIN_<DD/TD>_BLEND_LAYER_CONTROL_0

DD: Offset: 0x96 | Byte Offset: 0x258 | Read/Write: R/W | Reset: 0x01000000 (0bxxxxxxx1000000000000000000000000)
TD: Offset: 0x116 | Byte Offset: 0x458 | Read/Write: R/W | Reset: 0x01000000 (0bxxxxxxx1000000000000000000000000)

Bit	Reset	Description
24	BLEND_BYPASS	<DD/TD>_BLEND_BYPASS: 0 = BLEND_ENABLE 1 = BLEND_BYPASS
23:16	0x0	<DD/TD>_K2
15:8	0x0	<DD/TD>_K1
7:0	0x0	<DD/TD>_WINDOW_LAYER_DEPTH

24.13.7 DC_<D/T>_WIN_<DD/TD>_BLEND_MATCH_SELECT_0

DD: Offset: 0x97 | Byte Offset: 0x25c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)
TD: Offset: 0x117 | Byte Offset: 0x45c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xx00x000x000)

Bit	Reset	Description
13:12	0x0	<DD/TD>_BLEND_FACTOR_DST_ALPHA_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = NEG_K1_TIMES_SRC 3 = K2
9:8	0x0	<DD/TD>_BLEND_FACTOR_SRC_ALPHA_MATCH_SELECT: 0 = ZERO 1 = K1 2 = K2
6:4	0x0	<DD/TD>_BLEND_FACTOR_DST_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K2 4 = K1_TIMES_DST 5 = NEG_K1_TIMES_DST 6 = NEG_K1_TIMES_SRC 7 = NEG_K1

Bit	Reset	Description
2:0	0x0	<DD/TD>_BLEND_FACTOR_SRC_COLOR_MATCH_SELECT: 0 = ZERO 1 = ONE 2 = K1 3 = K1_TIMES_DST 4 = NEG_K1_TIMES_DST 5 = K1_TIMES_SRC

24.13.8 DC_<D/T>_WIN_<DD/TD>_BLEND_ALPHA_1BIT_0

DD: Offset: 0x99 | Byte Offset: 0x264 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)
TD: Offset: 0x119 | Byte Offset: 0x464 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	<DD/TD>_BLEND_WEIGHT1: This is used only for 1-bit alpha and alpha value of 1.
7:0	0x0	<DD/TD>_BLEND_WEIGHT0: This is used only for 1-bit alpha and alpha value of 0.

24.14 WINBUF_DD/TD Registers

Note: *In the register and field names, <D/T> indicates D or T, and <DD/TD> indicates DD or TD, depending on the window selection.*

24.14.1 DC_<D/T>_WINBUF_<DD/TD>_START_ADDR_0

Window DD/TD Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally, START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see “For linear address mode” under Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, the starting address of a window is calculated as follows by hardware.
  - non-YUV-planar modes

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-YUV-planar mode

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    where denom1/denom2 are equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode

  Image surface can only aligned to multiples of 256, thus the following restrictions.
  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be even in yuv planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset.
  - ADDR_V_OFFSET needs to be multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats
- For linear address mode

  Image surface can be aligned 2, 4, or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16. When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)
  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.
  - For 16-bpp formats,

    (START_ADDR+H_OFFSET) needs to be a multiple of 2.
  - For 32-bpp formats,

    (START_ADDR+H_OFFSET) needs to be a multiple of 4.
  - For YUV planar formats:
    - BUF_STRIDE, UV_BUF_STRIDE:
    - BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE is a multiple of 8, UV_BUF_STRIDE be a multiple of 4.
    - LINE_STRIDE, UV_LINE_STRIDE:

      LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

      LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

- ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset.
- ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset

- Memory allocation for non-host triggered case:

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

DD: Offset: 0xc0 | Byte Offset: 0x300 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x140 | Byte Offset: 0x500 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<DD/TD>_START_ADDR: Window DD/TD Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

24.14.2 DC_<D/T>_WINBUF_<DD/TD>_ADDR_H_OFFSET_0

Window DD/TD Horizontal address offset

DD: Offset: 0xc6 | Byte Offset: 0x318 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x146 | Byte Offset: 0x518 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<DD/TD>_ADDR_H_OFFSET: Window DD/TD Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

24.14.3 DC_<D/T>_WINBUF_<DD/TD>_ADDR_V_OFFSET_0

Window DD/TD Vertical address offset

DD: Offset: 0xc8 | Byte Offset: 0x320 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x148 | Byte Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<DD/TD>_ADDR_V_OFFSET: Window DD/TD Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of Y plane. The vertical coordinate of U/V plane is derived by hardware.

24.14.4 DC_<D/T>_WINBUF_<DD/TD>_UFLOW_STATUS_0

Window DD/TD FIFO Underflow Status Register

DD: Offset: 0xca | Byte Offset: 0x328 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x14a | Byte Offset: 0x528 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

24.14.5 DC_<D/T>_WINBUF_<DD/TD>_START_ADDR_HI_0

Window DD/TD Higher 32 bits of Start Address

Makes the start address 64 bits by adding the higher 32 bits below.

DD: Offset: 0xcd │ Byte Offset: 0x334 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
TD: Offset: 0x14d │ Byte Offset: 0x534 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	<DD/TD>_START_ADDR_HI: Window DD/TD Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

24.14.6 DC_<D/T>_WINBUF_<DD/TD>_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

If <DD/TD>_UFLOW_CYA is set, the underflow recovery logic for simple windows is bypassed and replaced with an overflow-pop logic present in Tegra 3 styled windows.

DD: Offset: 0xe4 │ Byte Offset: 0x390 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
TD: Offset: 0x164 │ Byte Offset: 0x590 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	<DD/TD>_UFLOW_CYA: 0 = DISABLE 1 = ENABLE
0	0x0	<DD/TD>_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

24.14.7 DC_<D/T>_WINBUF_<DD/TD>_UFLOW_DBG_PIXEL_0

DD: Offset: 0xe5 │ Byte Offset: 0x394 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)
TD: Offset: 0x165 │ Byte Offset: 0x594 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	<DD/TD>_UFLOW_DBG_PIXEL

24.14.8 DC_<D/T>_WINBUF_<DD/TD>_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra X1 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the VBlank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for the MC, it has a high potential to starve out other clients.

SPOOL_UP_EDGE:- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- If SPOOL_UP_EDGE is programmed to NEGEDGE, it ensures that if any act_req is sent during the time vcounter = 0 (at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value.

  The downside of programming SPOOL_UP_EDGE to NEGDGE is 1 line clock worth of spool up is lost. This is required based upon Tegra compatibility.

- If SPOOL_UP_EDGE is programmed to POSEDGE, the memfetch starts fetching at the very beginning of the frame start pulse (1 line clock wide) and gives maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above.

DD: Offset: 0xe7 | Byte Offset: 0x39c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
TD: Offset: 0x167 | Byte Offset: 0x59c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	<DD/TD>_SPOOL_UP_DURATION
1	0x0	<DD/TD>_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE
0	0x0	<DD/TD>_SPOOL_UP_CTRL. 0 = MAX 1 = PROGRAMMABLE

24.14.9 DC_<D/T>_WINBUF_<DD/TD>_LATENCY_THRESHOLD_0

DD: Offset: 0xe9 | Byte Offset: 0x3a4 | Read/Write: R/W | Reset: 0x0000ffff (0b00x00000000000001111111111111111)
TD: Offset: 0x169 | Byte Offset: 0x5a4 | Read/Write: R/W | Reset: 0x0000ffff (0b00x00000000000001111111111111111)

Bit	Reset	Description
31	DISABLE	<DD/TD>_RDY4LATENCY_THRESHOLD_ENABLE: 0 = DISABLE 1 = ENABLE
30	DISALLOW	<DD/TD>_RDY4LATENCY_SPOOLUP_CTRL: 0 = DISALLOW 1 = ALLOW
28:16	0x0	<DD/TD>_RDY4LATENCY_SPOOLUP_DURATION
15:0	0xffff	<DD/TD>_RDY4LATENCY_THRESHOLD

24.14.10 DC_<D/T>_WINBUF_<DD/TD>_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

DD: Offset: 0xea | Byte Offset: 0x3a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx00xxxxxx0)
TD: Offset: 0x16a | Byte Offset: 0x5a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx00xxxxxx0)

Bit	Reset	Description
9	0x0	<DD/TD>_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	<DD/TD>_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
0	0x0	<DD/TD>_UNDERFLOW_LINE0: Underflow of line0. 0= No underflow. 1= line0 underflowed

24.14.11 DC_<D/T>_WINBUF_<DD/TD>_MEMFETCH_CONTROL_0

Memfetch Control Register

DD: Offset: 0xeb | Byte Offset: 0x3ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)
TD: Offset: 0x16b | Byte Offset: 0x5ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	<DD/TD>_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	<DD/TD>_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

24.14.12 DC_<D/T>_WINBUF_<DD/TD>_OCCUPANCY_THROTTLE_0

DD: Offset: 0xec | Byte Offset: 0x3b0 | Read/Write: R/W | Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)
TD: Offset: 0x16c | Byte Offset: 0x5b0 | Read/Write: R/W | Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	<DD/TD>_OCCUPANCY_MAX_THRESHOLD
0	0x0	<DD/TD>_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

24.14.13 DC_<D/T>_WINBUF_<DD/TD>_SCRATCH_REGISTER_0_0

DD: Offset: 0xed | Byte Offset: 0x3b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)
TD: Offset: 0x16d | Byte Offset: 0x5b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	<DD/TD>_SCRATCH_REGISTER_0: Scratch register 0

24.14.14 DC_<D/T>_WINBUF_<DD/TD>_SCRATCH_REGISTER_1_0

DD: Offset: 0xee | Byte Offset: 0x3b8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)
TD: Offset: 0x16e | Byte Offset: 0x5b8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	<DD/TD>_SCRATCH_REGISTER_1: Scratch register 1

CHAPTER 25: COLOR DECOMPRESSION ENGINE

The color decompression engine (CDE) provides for color decompression which supports end-to-end data compression from surface render to display. The CDE resides between the GPU surface consumers (Display Controller or VIC) and memory.

The CDE allows GPU surface consumers to read compressed surfaces. The CDE converts the uncompressed surface requests into compressed surface requests, decompresses the data, and returns it to the GPU surface consumer uncompressed.

Refer to Chapter 24: Display Controller and Chapter 15: Video Image Compositor (VIC) for more information.

25.1 Features

CDE features include support for:

- Decompression of 4BPP surface colors
- Decompression ratios of 1:2, 1:8
- Arithmetic decompression (2:1)
- Reduction decompression (8:1)
- ZBC (Zero Bandwidth Clear)
  - Single ZBC color per surface
- Up to 8 surfaces
  - Compression bit buffer sized to support up to 2:1 downscaling for 8 surfaces
- Compression bit prefetching
  - Hide latency of fetching compression bits then fetching compressed data
  - Amortized over multiple scan lines
  - 128B fetch granularity
- Decompression bypass
  - Override requests to compressed surfaces
- Up to 4Kx4K pixel surface support
  - Maximum panel resolution – 4096 x 2560
- Surface Occlusion support
  - Up to 8 surfaces
- Immediate Flip support
  - Flip delayed to align with ROP tile boundary scan line

25.2 Functional Description

25.2.1 CDE Overview

The design point of CDE illustrated in the figure below is as follows:

- The CDE is a unit that exists between the compressed surface consumer and memory controller
  - Either between display and MCCIF or VIC and MCCIF
- The surface consumer will make requests for uncompressed data

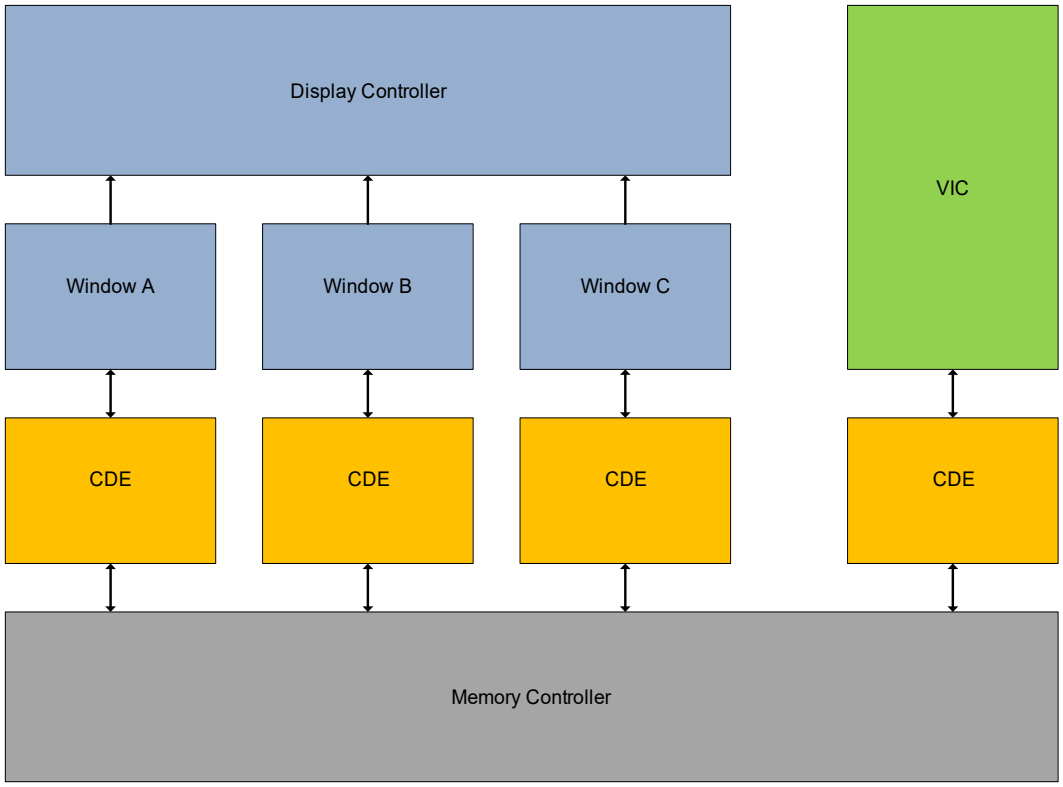
- The CDE determines whether the data is compressed.
  - If it is compressed, the CDE fetches the corresponding compressed data, decompresses it, and sends it to the GPU surface consumer
  - If it is uncompressed, the CDE passes the request and response through unmodified

The overall strategy for supporting decompression is as follows:

- GPU renders compressed surface
- Flush L2 cache and compression bit cache (CBC)
- Software running on GPU reformats CBC bits to allow for surface traversal either in horizontal or vertical fashion.
- Software updates registers in GPU surface consumer with respect to the new compressed surface
  - CDE registers part of surface fetch engine (i.e., VIC or display)
- GPU surface consumer reads from the compressed surface and CDE decompresses the surface as it returns the data

Note: *Since a surface flush and reformatting of CBC bits are required the surface must be at least double buffered. The GPU cannot render to a surface being displayed since display does not have access to the GPU L2 cache.*

Figure 100: CDE Design Point



25.3 CDE Registers

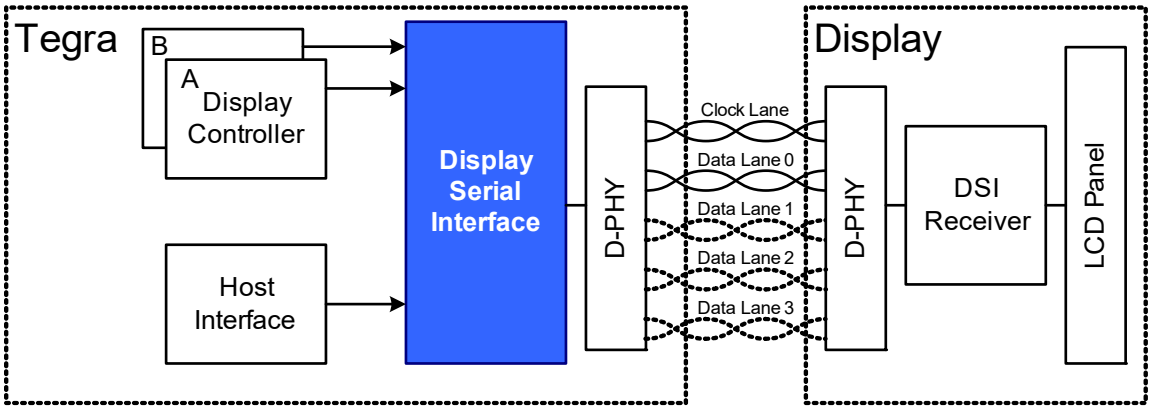
Refer to the following subsections in Chapter 24: Display Controller in this TRM for definitions of the CDE registers:

- WINBUF_A CDE Registers
- WINBUF_B CDE Registers
- WINBUF_C CDE Registers

CHAPTER 26: DISPLAY INTERFACES: MIPI-DSI

The Display Serial Interface (DSI) is a MIPI standard serial bitstream that provides a low pin-count interface to a display panel. DSI reduces both package pin-count and I/O power consumption compared with parallel solutions. It transfers pixel data from either one of the display controllers (internal to the Tegra® X1 device) to an external third-party LCD module. The physical positioning of the DSI module in relation to other units/devices in the system is shown below.

Figure 101: System Block Diagram of Single DSI Unit



DSI is a replacement for the MIPI DPI and DBI standards and follows the use cases of these interfaces. From a data transport point of view, MIPI DPI is an interface similar to a traditional raster-based isochronous display interface, and DBI is an asynchronous packet based transfer mechanism. DSI behaves like MIPI DPI when in Video Mode, and implements a Command Mode to handle the DBI interface. Any implementation must implement both these modes of operation.

26.1 Features

- Two DSI controllers, each drives the data on the respective D-PHY lanes
- Each DSI interface has 1 clock lane and 4 data lanes
- Maximum 1.5 GHz HS transmit rate
- Maximum 10 MHz LP receive rate
- Supports up to 8 PHY lanes with each DSI instance/channel supporting 1, 2, 3, 4 data lanes in Ganged mode configuration
- Video Mode with display controller as master
- Command Mode with host and/or display controller as masters
- Maximum Video Mode resolution is:
  - Uncompressed: 3480x1920, 60 fps at 24 bpp
  - Compressed: 4096x2304, 60 fps with 3:1 compression (8 bpp)
- Maximum Command Mode packet length is 1920 words
- Ganged Mode (Odd-Even/Left-Right)
- Dummy/Overlap mode

26.2 Functionality

26.2.1 Display Controller Interface

The interface between the display controller and the DSI unit consists of pixel data, Sync data, and a Line-Type code.

DSI captures the state of the line type code every line. This information is then used to look up a pre-programmed packet sequence that corresponds to that line type. There are several different line types to select from.

26.2.2 Packet Poster

The packet poster determines which packets will be sent on what video line when the DSI is in any of the 3 video modes and is being controlled by the display controller. The functions that this sub-unit performs are:

- Line type look-up
- Packet sequence generation
- Pixel data / packet stream integration

There are currently 6 valid line types that can be encoded on the 3 bits from the display controller. The other 2 line types are reserved for future use. The line type coding is shown in the following table.

Table 128: Line Type Descriptions

Line Type	Description
0	Blank line starting with VS
1	Blank line starting with VE
2	Blank line starting with HS
3	Active line
4	First blank line after active
5	First active line

The packet sequences corresponding to each line type are held in a pair of 32-bit registers. Each register holds three packet descriptors, and each packet descriptor has the following information:

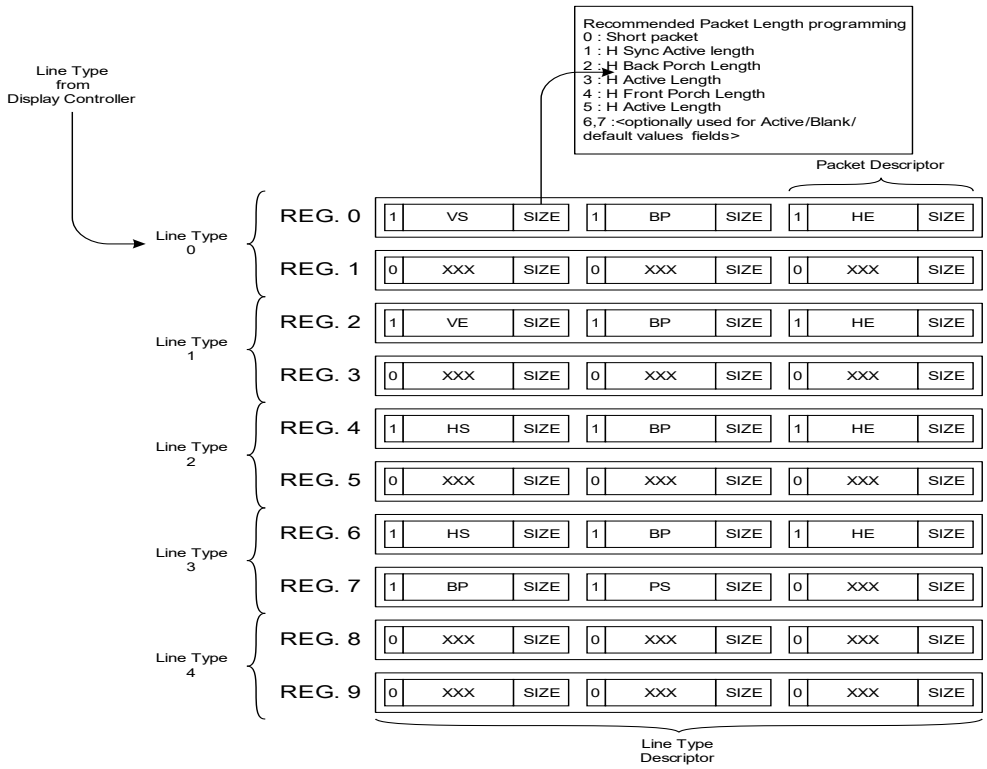
- Valid bit
- 6-bit packet ID
- 3-bit length index

The valid bit simply states whether or not a packet should be generated from that descriptor. A zero valid bit implies that no more packets in this packet sequence should be generated. Thus, the valid bit acts like a null terminator for the list of packet descriptors.

The packet ID is simply the MIPI DSI packet ID value. Software can therefore program any sequence of packets required for that line type.

Rather than have a 16-bit packet length field for every packet descriptor, the length is replaced by a 3-bit index which addresses the full 16-bit length for that packet type. This length is held elsewhere in separate length registers. A complete picture of the packet description data structures is shown in the following figure. This diagram shows typical packet mnemonics for the packets to be generated for video mode with sync pulses. By changing the programming of the packet sequences, any of the video modes, including burst mode, can be realized.

Figure 102: Line Type Look-up and Packet Sequence Descriptors



26.2.3 Host Interface

While many displays use a raster-based packet stream, in some applications it is more efficient to have a frame store in the display device itself and to only update the contents of the frame store when the required pixels have changed. For these types of displays, there is a software-accessible interface for sending command packets using the DSI command mode operation. This mode should also be used to access control and status registers within the display device.

The Host interface consists of a small FIFO for holding packet information and a special register for writing data into the FIFO. In this way, several packets may be queued up by software and sent in one HS burst. This is more efficient that sending the packets one at a time.

When very long sequences of packets are required to be sent – for example, when the host interface is emulating video mode, or where a large amount of data needs to be sent to the display device in one packet – it is possible for the host to use the large video line-store FIFO.

26.2.4 Clocking

The clocks in the DSI include:

- Host Clock = 300 MHz {Maximum frequency}
- Application/Lane Management Byte Clock = function of lanes, pixel clock, pixel depth. 10 to 187.5 Mbps
- Fixed LP receive clock = 20 to 216 MHz
- Lane Bit clock = 40 to 750 MHz differential clock, DDR data (80 to 1500 bps).
- DSI LP clock source is PLLP.

More details on pixel clock/byte_clock selection are explained in the next section.

Pixel Clock / DSI Byte Clock Ratios

When running the DSI interface in one of the two non-burst video modes, the relationship between the DSI D-PHY bit-rate clock, the DSI byte-rate lane clock, and the display controller pixel clock must be exact. This ensures that precise raster timing information is conveyed to the display peripheral.

There is always a fixed ratio of 8:1 between the D-PHY bit clock and the lane management layer byte clock since there are 8 bits in each byte that are serialized. Therefore, only relationships between pixel clock and byte clock and between pixel clock and bit clock will be discussed.

The relationship between pixel clock and byte clock is shown in the table below. As can be seen, for some modes, these ratios are not simple powers of 2.

Table 129: Pixel Clock: Byte Clock for Various Modes

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
1	1:2	4:9	1:3	1:3
2	1:1	8:9	2:3	2:3
3	3:2	12:9	1:1	1:1
4	2:1	16:9	4:3	4:3
5	5:2	20:9	5:3	5:3
6	3:1	8:3	2:1	2:1
7	7:2	28:9	7:3	7:3
8	4:1	32:9	8:3	8:3

Because the pixel clock is effectively derived from the D-PHY bit clock, it can be more instructive to look at the number by which you must divide the bit clock to get the correct pixel clock. This information is shown in the table below. Here, the numbers look reasonable except for the problematic cases of 16bpp over a 3-lane link and 18 bpp (packed) over a 4 lane link. For these two situations, you cannot simply divide the bit rate clock by some integer to arrive at the pixel clock. Moreover, if you need a 50:50 duty cycle for the pixel clock, then you will need to toggle the pixel clock every N/2 bit clock cycles, where N is the number in the table below. However, if you do this, 18 bpp (packed) over 2 lanes also becomes a problem.

Table 130: Value to Divide Bit Clock by to Get Pixel Clock

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
1	16	18	24	24
2	8	9	12	12
3	16 / 3	6	8	8
4	4	9 / 2	6	6
5	16/5	18/5	24/5	24/5
6	8/3	3	4	4
7	16/7	18/7	24/7	24/7
8	2	9/4	3	3

Table 131: Value of Shift Clock Divider to Get Pixel Clock (Non-Burst Mode

Lanes	Data Format (16 bpp)	Data Format (18 bpp)	Data Format (24 bpp)
1	14	16	22
2	6	7	10
3	3.333334	4	6
4	2	2.5	4

Table 131: Value of Shift Clock Divider to Get Pixel Clock (Non-Burst Mode

Lanes	Data Format (16 bpp)	Data Format (18 bpp)	Data Format (24 bpp)
5	1.6	1.8	2.4
6	0.666667	1	2
7	1.14285714	1.285714286	1.714285714
8	0	0.25	1

Note: *The fractional values in the above table cannot be supported through configuration fields because of a possible limitation in programming the shift divider for deriving the pixel clock from the byte clock in non-burst mode, that is, the SHIFT_CLK_DIVIDER field of the DC_DISP_DISP_CLOCK_CONTROL_0 field. This limitation occurs only in non-burst video mode.*

To resolve these issues, derive the display controller pixel clock from a separate PLL and then lock that PLL to the DSI D-PHY PLL using a phase comparator and the appropriate divide ratios obtained from the tables.

DSI PLL – Clock Mux

In the Tegra X1 device, the DSI implements a clock mux for the clock selection of the write clock domain to the line buffer and its upstream path in the DSI.

A single DSI instance receives a pixel data stream from either of the display controller heads. The single bit register field dsi_vid_src in the dsi_clk domain selects the active display head. The selected display pixel clock provides the necessary clocking to the front end of the DSI.

In cases where the Host transmission uses the line buffer, dsi_clk can provide the necessary clocking using the single bit register field pkt_wr_fifo_sel.

26.3 Modes of Operation

The DSI operates in two transmission modes: Video and Host/Command

26.3.1 Video Mode

Communication with the peripheral is by isochronous data transfer similar to a typical video interface. There is no Bus Turn Around permitted in Video Mode, though there is a mandatory period of LP operation at least once per frame. There are three different sub-modes of Video Mode. These are outlined below:

- Non-Burst Mode with Start and End

  In this mode, the DSI must match the timing of a traditional video raster as closely as possible. This means all information about the start and end of parameters like vertical and horizontal sync, front and back porches must be conveyed to the receiving peripherals via the timing of the transmission of the High-Speed sync packets.

- Non-Burst Mode without Start and End

  This is like Non-Burst Mode with Start and End, except that the Sync-Active and Sync-End packets are not sent. Pixels must still be delivered at the correct rate.

- Burst Mode

  Only the timing of Sync-Start packets is required to be accurately conveyed in this mode. The data rate of the pixel data is arbitrary and is typically higher than the pixel rate of the peripheral.

26.3.2 Host/Command Mode

Communication with the peripheral is by asynchronously timed and variable length packets. The packets may contain video data (Display Controller/Host) or control data. In Command Mode, return (read) data can be requested from the peripheral. The DSI will issue a Bus Turn Around (BTA) request and relinquish control of the bus to the peripheral.

26.4 FIFO Buffers

26.4.1 Video Mode Operation

26.4.1.1 Writing Data

The following packets are written into the large data FIFO based on the various triggering events described.

Table 132: FIFO Trigger Events

Trigger Event	Packet
Leading edge of Vertical Sync.	VSync Start
Trailing edge of Vertical Sync	VSync End
Leading edge of Horizontal Sync, *unless* simultaneous with leading or trailing edge of vertical sync	HSync Start
Trailing edge of Horizontal Sync	HSync End
Immediately following VS, HS, VE or HE packets. Packet Word Count determined by display controller timing parameters.	Blanking packet
Immediately following blanking packet	Pixel Stream packet (16, 18, or 24 bits/pixel)

26.4.1.2 Reading Data

The initiation of the reading of the FIFO is triggered by a delayed version of the horizontal sync pulse from the display controller. The delay should be sufficient that the FIFO will not completely drain as reading of the FIFO continues. However, the delay should not be so long as to cause the FIFO to overflow with data.

26.4.2 Command Mode Operation

26.4.2.1 Writing Data from the Display Controller

Upon receipt of the leading edge of Vertical Sync, the display controller sends whatever DCS command packets are required to initialize the display to receive pixels. Then, as pixels start to arrive from the display controller, a DCS Long Write packet is sent – one per line – which contains the pixel data for that line. No synchronization or blanking packets should be sent. Data packets continue to be sent until the end of the active period. Like video mode packets, there is no need to calculate ECC or CRC words, as this is performed by the hardware on the other side of the data FIFO.

The Packet Sequence registers behave in the same manner as they do for Video Mode packet generation with the following exceptions:

- For Long packets, a DCS command ID byte must be inserted as the first byte of the data payload of the packet, ahead of the pixel data. The packet header Word Count must be 1 more than the number of bytes in the pixel data to take into account this extra byte. It is the responsibility of software to make sure the packet length is programmed correctly.
- For Line Type 4, any data contained in the Host Data FIFO should be appended to the end of any packets defined in the Packet Sequence for Line Type 4. In this way, the Host may send DCS commands to the Display Peripheral while the Display Controller is sending pixel information to the Display Peripheral.

26.4.2.2 Writing Data from the Host

The host should choose which data FIFO – small or large – to use prior to sending packet information. The large data FIFO is only available if the display controller is not currently using it. The software is responsible for constructing whatever packets are

required for the desired operation and should be written into the FIFO via the host. There is no need for the software to compute ECC or CRC words, because this is performed by the hardware on the other side of the data FIFO.

26.4.2.3 Reading Data

The initiation of the reading of data from the FIFO is triggered by a FIFO threshold. Once the threshold is reached, the D-PHY starts to drain the FIFO, and the packets are sent to the display. The threshold should be set such that there is no danger of either FIFO overflow from excessive data from the host or display controller, or FIFO underflow caused by a lack of sufficient data in the FIFO.

26.5 Programming Guidelines

The packet timing diagrams in this section use the following key:

Figure 103: Video Mode Timing Diagram Key

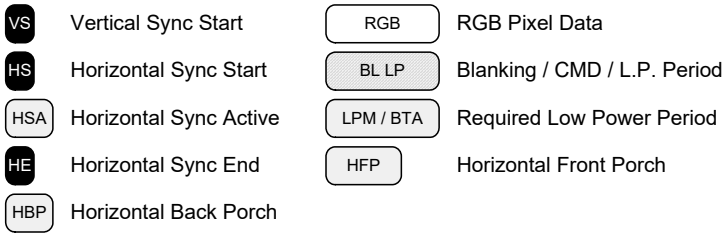


Table 133: Timing Diagram Parameter List

Parameter	Description
tHBP	Horizontal Back Porch period
tHACT	Horizontal Active period
tVBP	Vertical Back Porch period
tVACT	Vertical Active period
tVFP	Vertical Front Porch period
tL	Total line period

26.5.1 Initialization

The start-up and shut-down procedures vary depending on what mode of operation is required. Examples of programming sequences for the main modes are given below.

26.5.1.1 Video Mode Start-up

All 3 video modes – Burst, Non-Burst and Non-Burst with Sync Ends –have the same start-up sequence with a slight variation between Burst and Non-Burst modes:

- Set up the clocks. This involves configuring and enabling the DSI PLL (PLLD). For Non-Burst and Non-Burst with Sync End modes, the Display Controller must also be programmed to take its pixel clock from the DSI PLL. For Burst Mode, the Display Controller can take its pixel clock either from the DSI PLL or from another clock source. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.
- Depending on the sub-mode, the various Packet Sequence registers and Packet Length registers must be programmed.
- Set the VID_TX_TRIG_SRC field in the DSI_CONTROL register to SOL. And program DSI_VID_SOURCE to select which display controller will source the video data.
- Enable the DSI

- Program the display controller to produce the raster size required
- Enable the display controller

When the display controller starts sending SOL and data, the DSI automatically starts creating and transmitting packets to the display device.

26.5.1.2 Command Mode Start-up

- Set up the clocks. Program and enable the DSI PLL (PLLD). In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register. Unlike the Video modes, the selection of the PLLD output clock frequency is essentially arbitrary. The clock frequency should be based on the expected data throughput and the requirements of the particular display device. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.
- Set the HOST_TX_TRIG_SRC field in the HOST_DSI_CONTROL registers to `IMMEDIATE`. Set any other state required.
- Enable DSI.

The DSI should now be ready to accept command packets written to the DSI_WR_DATA register.

26.5.2 Video Mode Programming

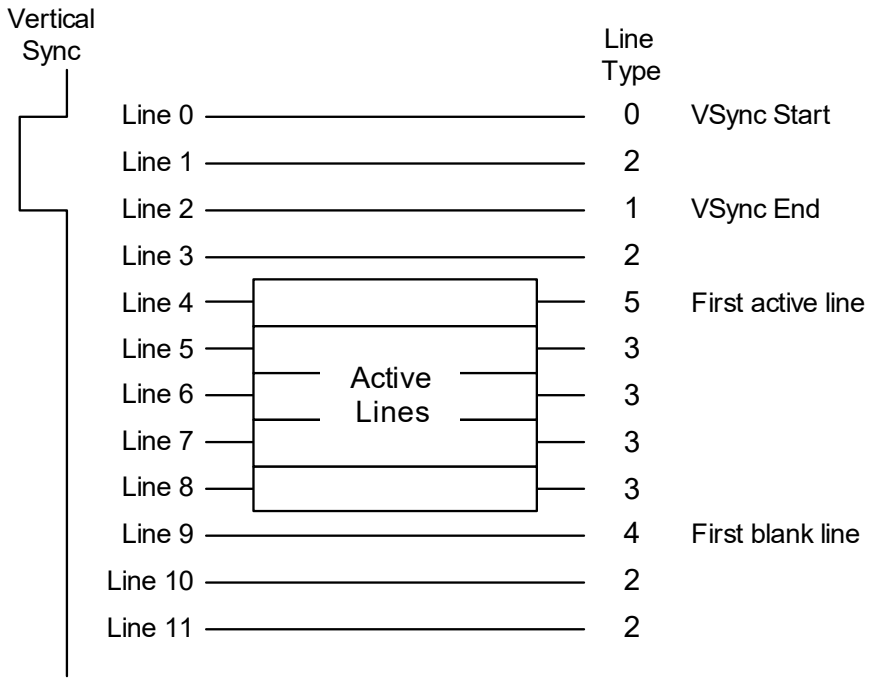
The programming model has been designed to accommodate current video mode sequences and potential future variations. Each line of video output is associated with a “line type”. This line type is automatically generated by the display controller hardware according to the following table.

Table 134: Recommended Line Types

Line Type	Description	Typical Raster Position
0	Blank line starting with VS	First line of the raster
1	Blank line starting with VE	Line corresponding to Vsync end.
2	Blank line starting with HS	Vertical blanking line
3	Active line starting with HS	Active line
4	First blank line after active	First blank line after active
5	First active line	First active line

The figure below illustrates a typical raster showing the relationship between various events in the raster and the line types.

Figure 104: Example Video Raster Showing Line Types



The line type acts like a pointer to a data structure containing the information about the "*packet sequence*" for that line type. The packet sequence information is actually held within a pair of registers. For example, the packet sequence for line type 0 is held in the pair of registers DSI_PKT_SEQ_0_LO and DSI_PKT_SEQ_0_HI. Each pair of packet sequence registers contains all the information required to generate up to 6 packets. This is sufficient to generate any packet sequence for any DSI video line. The information stored for each packet is shown in the following table:

Table 135: Packet Sequence Description Fields

Field	Number of bits	Description
PKT_*_SIZE	3	Pointer to packet size register
PKT_*_ID	6	Packet ID as defined in the MIPI DSI specification.
PKT_*_EN	1	Enable. Determines which packets are active.

Using the same pair of registers as an example, the table below shows how all six packet descriptors fit into the pair of 32-bit packet sequence registers.

Table 136: Packet Sequence 0 Registers

Register	Field	Bits	Description
DSI_PKT_SEQ_0_LO	PKT_00_SIZE	2:0	Packet 0 Size
	PKT_00_ID	8:3	Packet 0 ID
	PKT_00_EN	9	Packet 0 Enable
	PKT_01_SIZE	12:10	Packet 1 Size
	PKT_01_ID	18:13	Packet 1 ID
	PKT_01_EN	19	Packet 1 Enable
	PKT_02_SIZE	22:20	Packet 2 Size
	PKT_02_ID	28:23	Packet 2 ID
	PKT_02_EN	29	Packet 2 Enable
	SEQ_0_FORCE_LP	30	End in LP state

Table 136: Packet Sequence 0 Registers

Register	Field	Bits	Description
DSI_PKT_SEQ_0_HI	PKT_03_SIZE	2:0	Packet 3 Size
	PKT_03_ID	8:3	Packet 3 ID
	PKT_03_EN	9	Packet 3 Enable
	PKT_04_SIZE	12:10	Packet 4 Size
	PKT_04_ID	18:13	Packet 4 ID
	PKT_04_EN	19	Packet 4 Enable
	PKT_05_SIZE	22:20	Packet 5 Size
	PKT_05_ID	28:23	Packet 5 ID
	PKT_05_EN	29	Packet 5 Enable

Finally, the SIZE field in the packet descriptor is used as a pointer to a 16-bit "*packet length*" field in the packet length registers. An indirect pointer is used rather than storing the packet length directly in the packet descriptor because the packet lengths are physically large – 16 bits – but there is a lot of re-use. There are only a few different lengths used in a typical raster layout. Thus, it is more efficient to hold the lengths in separate length registers and then to simply refer to them with a short pointer. This allows the storing of 36 packet descriptors in only 6 pairs of packet sequence registers.

Note: *One of the packet length registers is reserved for short packets. If a* `SIZE` *field in the packet descriptor is programmed to 0, this packet is likely a short packet. Short packets are fixed in length to 4 bytes including the packet header byte and ECC byte. In the context of video mode, they are essentially reserved for timing packets. All other packet length registers can be used to determine the length of any associated long packet.*

The `SEQ_0_FORCE_LP` bit is used to determine if the link should be placed in the LP state at the end of the packet transmission. In Burst Mode operation, the link always drops back into LP state at the end of the HS packet transmission on a line. However, for Non-Burst Modes, the HS transmission may continue to the next line. It is important the that the hardware state machine that generates packets not attempt to go to the LP state, but should instead prepare for the next sequence of packets associated with the next video line and keep the line in the HS transmission state.

The packet length registers are shown in the table below. The packet size pointer in the packet descriptor can point to any of the available length registers – with the one exception of short packets, whose `SIZE` field must point to length register 0. It is recommended that the suggested packet length assignment for various length registers is followed to reduce confusion when debugging packet descriptors and packet sequences.

Table 137: Packet Length Registers and Assignments

Register	Field	Suggested Assignment
DSI_PKT_LEN_0_1	LENGTH_0	Must be used for short packets
	LENGTH_1	Horizontal sync active length (HSA)
DSI_PKT_LEN_2_3	LENGTH_2	Horizontal back porch length (HBP)
	LENGTH_3	Horizontal active length (ACTIVE)
DSI_PKT_LEN_4_5	LENGTH_4	Horizontal front porch length (HFP)
	LENGTH_5	–Horizontal active length (ACTIVE)
DSI_PKT_LEN_6_7	LENGTH_6/7	<EOTP/other packet IDs>

When programming the length registers, remember that these contain byte counts, not pixel counts. Therefore, the DSI pixel format, number of DSI lanes in use and the finite (non-zero) size of the packet header and CRC words should be taken into account when programming these values – especially for blanking packets since all of these parameters affect the number of bytes that should be used to emulate a specific blanking time. The equations required to determine the byte count in the packets are given in the examples that follow. The identification of the length given in the tables and equations has been included in three video mode examples so that the values in the examples can be easily tied to the registers.

26.5.2.1 SOL Delay Programming

In order to ensure that the pixel FIFO from display to DSI does not underflow when in video mode, it is necessary to delay the start of packet generation by the DSI by a fixed amount from the arrival of the SOL signal from the display. This is especially true of Burst-Mode, because the DSI byte clock can be very much faster than display pixel clock. If no delay is applied, the DSI will very quickly consume all the pixels in the FIFO and the FIFO will underflow. The SOL_DELAY registers are used to set this delay.

Non-Burst Mode

In non-burst mode, the rate at which the DSI consumes pixels is the same as the rate at which the display module produces them, so it is merely sufficient to ensure that enough pixels have been fed into the FIFO to overcome the internal latency. This internal latency is approximately 8 pixel clock cycles. However, `SOL_DELAY` is programmed in DSI byte clocks, so the pixel format and the number of lanes being used should be taken into account when programming this value. Contact your NVIDIA FAE for details on the relationship between display pixel clock and byte clock. These ratios are then used to determine the value of `SOL_DELAY` as follows (these equations are applicable for DCS in Non-burst configuration as well):

$$SOL\_DELAY = (((Sol2VldDly) + 14) * F_{DSI} / F_{pixel}) + FifoLatency$$

$$FifoLatency = Ceil\ (2 * (F_{DSI} / F_{pixel})) + 6$$

Sol2VldDly = Hsync start to Pixel valid delay

Where F_{DSI} and F_{pixel} are the DSI byte and pixel clocks, respectively.

Note: SOL toggle to valid assertion delay assumed to be fixed at 4 pixel clocks, in case the delay exceeds 4. The sum of additional clock cycles shall be added to the SOL delay equation.

Burst Mode

In Burst mode, not only must the pixel format, number of lanes, and the DSI / pixel clock ratio be taken into account, but also the horizontal back porch time (including Hsync) and the Horizontal active time.

The below equations are applicable for DCS in burst configuration as well.

So, for Burst mode:

- In case of End of Transmission packet, i.e., Eotp enabled, set Eotp = 1
- When the HFP packet is not transmitted, additional latency, i.e., AddLatency=20
- SOL toggle to valid assertion delay assumed to be fixed at 4 pixel clocks, in case the delay exceeds 4. The sum of additional clock cycles shall be added to SOL delay equation.
- When EOTp packets are enabled:

  $$SOL\_DELAY = (((Sol2VldDly + 6 + T_{active} + 1) * F_{DSI} / F_{pixel}) -$$

  $$((T_{active} + 1) * F_{DSI\_NB} / F_{pixel\_NB}))) + FifoLatency + AddLatency$$
- When EOTp packets are not enabled:

  $$SOL\_DELAY = (((Sol2VldDly + 6 + T_{active}) * F_{DSI} / F_{pixel}) -$$

  $$((T_{active}) * F_{DSI\_NB} / F_{pixel\_NB}))) + FifoLatency + AddLatency$$

  That is,

  $$FifoLatency = Ceil\ (2 * (F_{DSI} / F_{pixel})) + 6$$

  Sol2VldDly = Hsync start to Pixel valid delay

Notes:

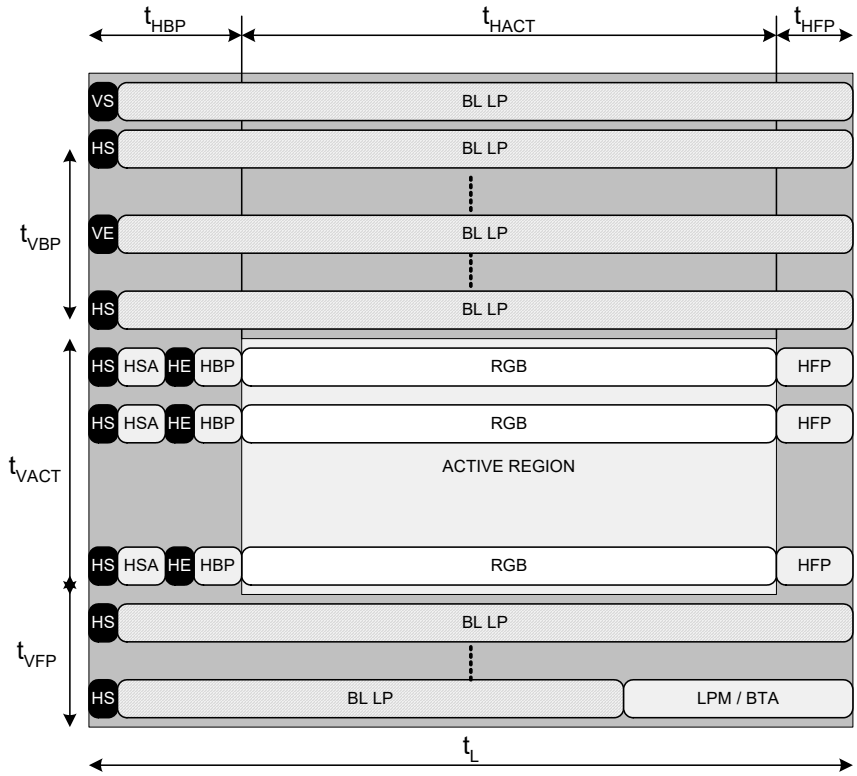
- *The ratio $F_{DSI\_NB} / F_{pixel\_NB}$ is determined by the clock ratio tables for Non-Burst mode, according to the pixel format used and the number of active lanes.*

- *If the overlap feature is enabled, the Tactive -> Effective Valid pixel length is inclusive of Overlap pixels.*

26.5.2.2 Non-Burst Mode with Start and End

This mode conveys traditional raster synchronization information across the link to the peripheral by sending both start and end sync packets.

Figure 105: Timing Diagram for Video Non-Burst Mode with Start and End



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period.

The figure below shows how a single line is constructed from multiple packets.

Figure 106: Non-Burst Mode with Start and End Packet Timing Detail

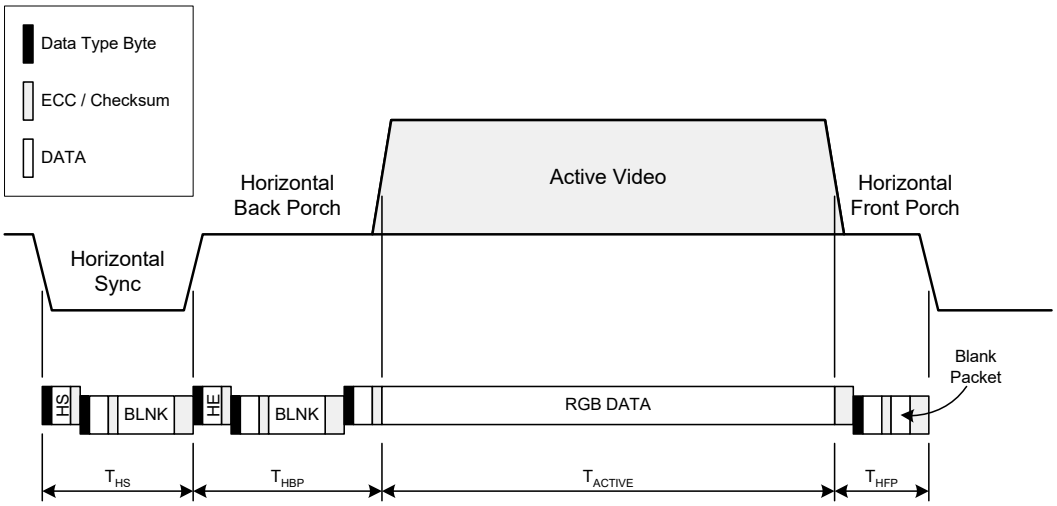


Table 138: Payload Size Table - Non-Burst Mode with Start and End

Packet	Payload Size (Bytes)
HSA	(T_{HS} * B) – 10
HBP	(T_{HBP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

B = 2, 2.25 or 3, depending on pixel format.

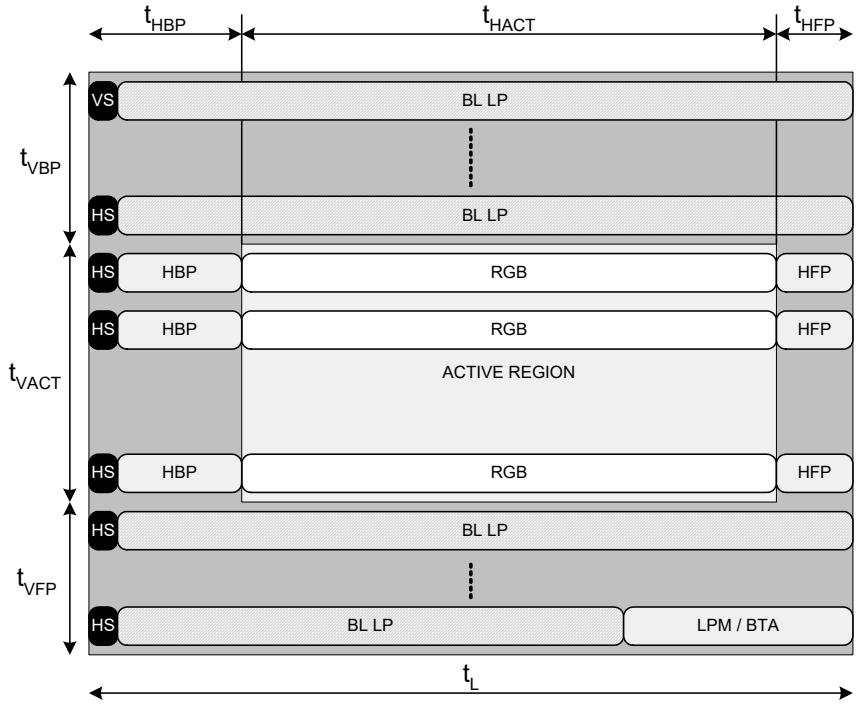
Table 139: Line Type Packet Sequences - Non-Burst with Sync Ends

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7								
1	VE	0	EOT	7								
2	HS	0	EOT	7								
3	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4
4	HS	0	EOT	7								
5	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4

26.5.2.3 Non-Burst Mode (without Ends)

This mode relaxes the requirement to mimic the generation of sync pulses and merely mandates that the start of the line is indicated and that the pixels appear at the same rate and in the same area of the raster as a tradition raster structure.

Figure 107: Timing Diagram for Video Non-Burst Mode



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period.

The figure below shows how a single line is constructed from multiple packets.

Figure 108: Non-Burst Mode Packet Timing Detail

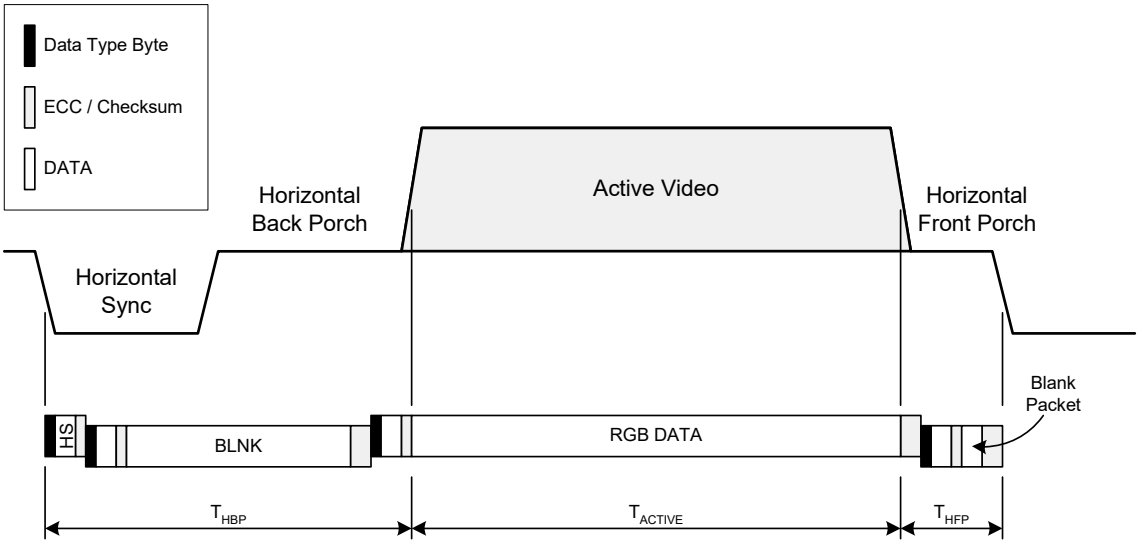


Table 140: Payload Size Table - Non-Burst Mode

Packet	Payload Size (Bytes)
HBP	$(T_{HBP} * B) - 14$
ACTIVE	$(T_{ACTIVE} * B)$
HFP	$(T_{HFP} * B) - 8$
1. B = 2, 2.25, or 3, depending on pixel format.	

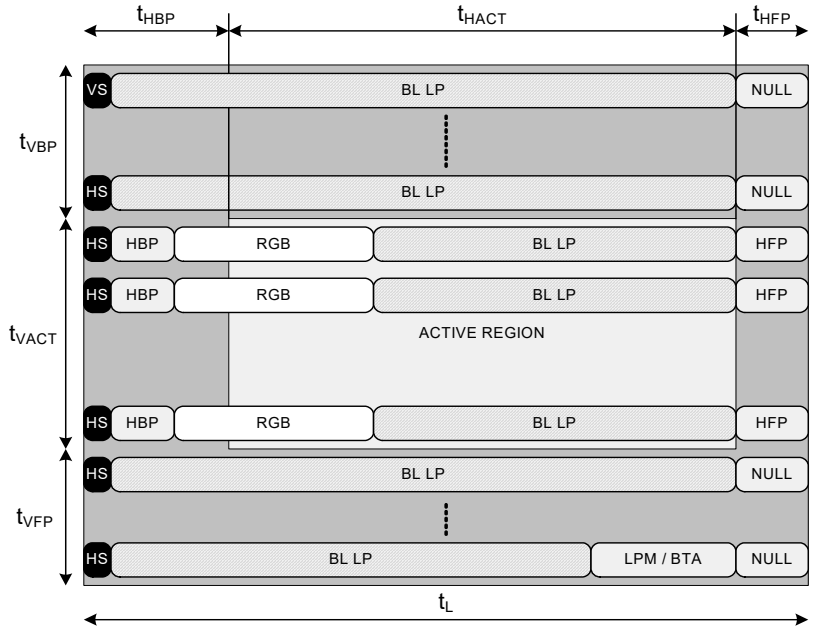
Table 141: Line Type Packet Sequences – Non Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7								
1	HS	0	EOT	7								
2	HS	0	EOT	7								
3	HS	0	BLNK	2	RGB	3	BLNK	4				
4	HS	0	EOT	7								
5	HS	0	BLNK	2	RGB	3	BLNK	4				

26.5.2.4 Burst Mode

In Burst Mode, the only attempt to match the raster structure is with the timing of the sync start events. The actual RGB pixel data is transmitted at whatever rate is convenient. This means that the HS, HBP, and RGB packets become compressed with respect to the timing of the underlying raster. This allows some period of idle time for each line that can be used for the transmission of other packets.

Figure 109: Timing Diagram for Video Burst Mode



During the Vertical Active period, packets on an individual line should be concatenated into a single HS transmission. However, unlike Non-Burst Mode, the bus should go idle – if possible – at the end of this transmission. This will allow additional non-video packets to be sent simultaneously with the video stream. The NULL and HFP packets may be optional. Check the latest MIPI DSI specification for clarification.

Table 142: Line Type Packet Sequences - Burst Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7				
1	HS	0	EOT	7				
2	HS	0	EOT	7				
3	HS	0	BLNK	2	RGB	3	EOT	7
4	HS	0	EOT	7				
5	HS	0	BLNK	2	RGB	3	EOT	7

Notes:

- *Burst mode always forces the LP packet field to LP mode. DSI_DSI_PKT_SEQ_0_LO/HI_0/1/2/3/4/5[SEQ_0_FORCE_LP] = 1'b1*
- *The register programming updates in video mode that impact the raster timing (DSI_DSI_PKT_SEQ_*_LO/HI_*, video mode control fields DSI_DSI_CONTROL_0) are assumed to be static with respect to the display controller. On-the-fly updates are not supported.*

26.5.2.5 Sequence while Switching the Modes/Updates to Raster Timing

This programming sequence must be followed when switching the modes/updates to raster timing:

1. Disable the DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1'b0)
2. Disable the DSI (DSI_LEG_EN disable)
3. Update the DSI configuration registers.
4. Enable the DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1'b1).

5. Enable the DSI (DSI_LEG_EN enable)

26.5.3 Command Mode Programming

There are two sources of command mode packet sequences:

- Display Controller
- Host Interface

Only one of these sources will be active at any particular time.

26.5.3.1 Command Mode from Host

In this mode of operation, all packets sent over the DSI interface are determined by software. There may be hardware assistance in the generation of error correction codes (ECCs) and cyclic redundancy checks (CRCs), but all other data is passed unaltered to the DSI physical layer.

Host Packet Writes

Packets are written to the hardware by writing to the DSI_WR_DATA register. The data written is passed into the DSI Host transmit FIFO. If the data is a packet header and the ECC_ENABLE field in the HOST_DSI_CONTROL register is set to ENABLE, then the MSBs of the 32-bit packet header word are replaced with a hardware-computed ECC byte prior to being written into the FIFO. If this field is set to DISABLE, then no action is taken by the hardware and the packet header is written unchanged.

If the packet is a long packet, then the packet payload should be written to the register after the packet header is written. If the CS_ENABLE field of the HOST_DSI_CONTROL register is set to ENABLE, then the hardware computes the check-sum and appends it to the packet information. If this field is set to DISABLE, then software must append the correctly computed check-sum to the packet data.

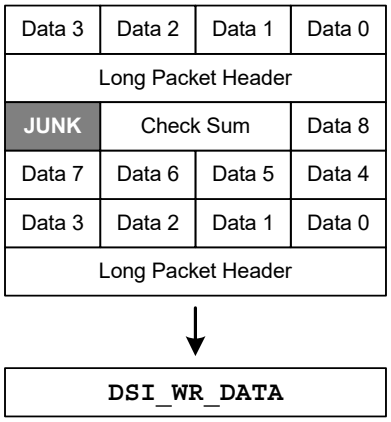
Rules:

1. Software should make sure packets are transmitted in their entirety and that once transmission starts, the FIFO contains enough data to finish a transition on a packet boundary. This can be achieved by only initiating a transmission – either explicitly or indirectly – once all data required for a transmission has been written to the transmit FIFO.
2. In Type-1 Display modules (i.e., the display device includes the full frame buffer to hold image data), the complete image data can be written to the frame buffer via the Host command mode. Software configures the register field to enable the Frame buffer selection for transfer of high-resolution image data.
   DSI_HOST_DSI_CONTROL[PKT_WR_FIFO_SEL]

   Note: *The maximum length of the long packet supported is 1920 words including header, payload, and CRC.*

3. Packets should be written such that the packet header is always written in one 32-bit word and is never split across writes. In other words, if the end of a packet does not fall on a 32-bit word boundary (if the payload has an odd length, for example), then the header for the next packet should not border the last packet, but should realign with the register. See the figure below for an example.

Figure 110: Packet Alignment for Writes to DSI_WR_DATA



Note: *For unaligned word transfers, the host transactions are split into multiple host transfers at the unaligned boundary.*

Host Packet Transmission

The source that controls when the Host write FIFO starts to drain (flush) is programmable. Selectable source is:

- Explicit FLUSH bit in a control register

  Immediate (HOST_TX_TRIG_SRC == IMMEDIATE**)**

As the name implies – if a '1' is written to the `DSI_HOST_TRIGGER` field of the `DSI_TRIGGER` register, then transmission of the data in the Host write FIFO will start immediately. It is recommended that all data required for transmission is written to the DSI_WR_DATA register prior to setting this bit.

26.5.3.2 Command Mode from Display Controller

This mode of operation allows the display controller to write pixel data packets to a DBI-like device that does not require data to be sent in an isochronous raster structure like a DPI device. The data coming from the display controller is sent in an isochronous manner, but the commands sent will be DCS control commands, rather than Video Mode timing and blanking packets.

Setup

There are six steps to operating in this mode:

1. Program the Peripheral display device using DCS commands via the Host Command interface. It is important to send the set_column_address, set_page_address, and set_pixel_format commands. These effectively define the area to which you will be writing pixels.
2. Program the DSI_INIT_SEQ_CONTROL and DSI_INIT_SEQ_DATA_* registers in the DSI interface with the appropriate values. Any commands required to be sent once per frame and *every* frame by way of setup prior to the pixel data being sent should be put in here.
3. Program the `DSI_DATA_FORMAT` field of the DSI_CONTROL register to the required pixel format. Note that `BIT18NP` should not be programmed – see the section below on pixel format restrictions.
4. Program the DSI_PKT_SEQ_* registers. Program the registers in the following way:
   - Packet Sequence registers for Line Types 0, 1, 2, and 4 should all be programmed with 0 in all the Packet Enable fields. In other words, there should not be any packets generated for these line types.
   - Packet Sequence registers for Line Types 3 and 5 should be programmed with a DCS Long Write packet.
   - To extend the support to transfer continuous HS packets (or) enable the Data lane in HS transmission during the entire active region (Line Type 3 and 5), additional Blank packets may be added to the packet sequence register. Refer to the table below for more information.

5. Program the DCS command ID register to have a write_start command associated with Line Type 5 (First line of active) and a write_continue command associated with Line Type 3 (all other active lines).
6. Enable the display. When Vertical syncs arrive from the display, the initialization sequence should be sent, and for every active line, the pixel data will be sent in a DCS Long Write packet.

Table 143: Line Type Packet Sequences for DCS Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	RGB	3	BLNK	4				
4	HS	0								
5	HS	0	RGB	3	BLNK	4				

Note: *Blank packets are optional and would be enabled to have the data lane in HS transmission during the active region, i.e., (Line Type 3 and 5).*

Pixel Format Restrictions

The MIPI DCS specification allows for up to 6 different pixel data formats to be transmitted in a write_start or write_continue command. These pixel formats are listed in the table below. The DSI supports 3 of these pixel formats.

The formats supported are also shown in the “supported” column of the table below. Do not set `BIT18NP` as a pixel format. There is no DCS equivalent of this format, so undefined behavior may result.

Table 144: DCS Pixel Format Support

DCS ID	DCS Format	Supported	Name
0	reserved	N/A	-
1	3bpp	N	-
2	8bpp	N	-
3	12bpp	N	-
4	reserved	N/A	-
5	16bpp	Y	BIT16P
6	18bpp	Y	BIT18P
7	24bpp	Y	BIT24P

Simultaneous Host Command Packets

It may be necessary, from time to time, to send a DCS command to the display peripheral in a “side-band” fashion while the display controller is continuing to send DCS write commands with pixel information. Thus Line Type 4 (first blank line) is reserved to indicate to the hardware when it should attempt to send any DCS command packets requested by software.

If it is desired to send a DCS command in this way, the DCS command packet should be written in its entirety (header, ECC, DCS command, payload, CSC) into the Host Command FIFO. The HOST_TX_TRIG_SRC field should then be set to `IMMEDIATE`.

The DSI hardware will then send the entire contents of the FIFO out on the DSI interface on the first line of blanking. Sending the data on this line guarantees the Host packet transmission will have enough time to complete before the next packet generated by the Display Controller is generated.

Notes:

- *Though the Host FIFO can accommodate a 256-byte packet, maxHost transfer size should not exceed as given in the below expression (units in byte clock count):*

(host_pkt_data_size < HOST_FIFO_size) && (host_pkt_data_size < (line_time – (sol_delay + min HS-LP-HS blanking time needed + 10)))

- *Host packets shall be aligned to word boundaries. In case host packets end on an unaligned word boundary, a maximum of 1 is supported for such consecutive packets transfers.*
- *Additional sideband data can be sent using Init sequence registers as well*
- *If the host transfer is enabled during the frame video blanking interval, the following sequence needs to be performed at the end of the frame:*

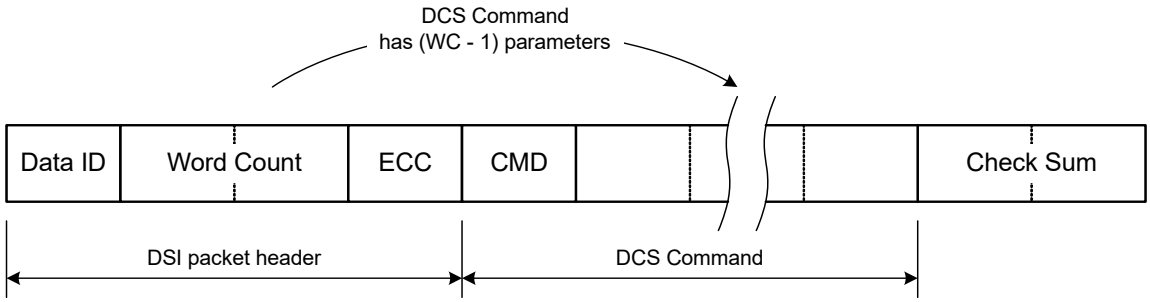
1. Wait for an end of frame event from the display controller.
2. Set the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1'b1.
3. Clear the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1'b0.
4. Continue with the subsequent configuration.

26.5.3.3 Display Command Set (DCS) Packets

DCS is a legacy control command set that is used to program various control registers present in typical LCD panels used in cell phones. Historically, the commands would be sent to the display over a MIPI DBI interface. Since MIPI DSI is meant to replace both DPI and DBI, it must also transport these control commands.

There are several DCS-specific commands in the MIPI DSI specification that can be used to send DCS commands to the Peripheral Display Device. However, the most useful is probably the DCS Long Write packet since it can be used to send commands with more than one parameter. The DCS command is embedded with an MIPI DSI Long Packet as follows: The DCS Command Byte and all the DCS Command Parameters (payload) are concatenated and sent in the Payload of the MIPI DSI Long Packet. The Word Count of the DSI packet conveys the total size of the DCS packet. As usual, a 2-byte CS footer is appended to the DSI Long Packet. The next figure below shows how a DCS Long Write packet is constructed.

Figure 111: DCS Command Placement in DSI Long Packet



26.5.3.4 16 bpp Byte Ordering in 16 bpp Format for Video/Command Mode

The DSI specification suggests a different byte order in the 16 bpp video and command mode data bit order, i.e., data packing in:

Video mode [16:0] -> {B[4-0],G[5-3]},{G[2-0],R[4-0]} (Default)

Command mode [16:0] -> {G[2-0],B [4-0]},{R[4-0],G[5-3]} (SwapEn)

The byte order can be configured through the DSI_DSI_CONTROL_0 register, DFMT_16BPP_SWAP_EN field. By default, the video mode order is selected and the command mode can be selected by the programing DFMT_16BPP_SWAP_EN to 1.

26.5.3.5 Frame Synchronization Between Identical Command Mode Displays

When stretching a single surface across multiple identical displays, frame synchronization can be achieved to avoid visual artifacts due to timing drifts across both the displays.

DCS command “adjust_vsync_timing” provides a mechanism for adjusting the command mode display vsync/frame sync timing with respect to another identical display. Vsync can be shifted, i.e., delayed/advanced by the specified number of scan lines.

26.5.3.6 Host Command Mode Byte Transmission

Command mode operation primarily involves sending the commands and data to the peripheral (the display device). The host processor indirectly controls activity at the peripheral by sending commands, parameters, and data to the display controller. Command Mode operation requires a bidirectional interface capabilities.

Command mode transfers are classified into,

- HS/LP mode

  Host command mode transmission can be enabled in High Speed and Low Power modes. Follow this programming sequence to configure the DSI to enable command mode transactions.

  - Configure DSI_DSI_CONTROL_0 [DSI_HOST_ENABLE]
  - LP/HS mode of transmission is selected by configuring the DSI_HIGH_SPEED_TRANS in the DSI_HOST_DSI_CONTROL_0 register.
  - Enable checksum/Ecc mode via DSI_HOST_DSI_CONTROL_0[CS_ENABLE], …, DSI_HOST_DSI_CONTROL_0[ECC_ENABLE].
  - Write the host data to the register DSI_DSI_WR_DATA_0. Once all the command packets are pushed into the FIFO, trigger the start of the transaction via the HOST_TX_TRIG_SRC field in the DSI_HOST_DSI_CONTROL_0 register.

- Raw Mode

  Host command mode transmission also supports RAW data bytes to be transferred to the display device. In Raw mode, the data written to the host is transferred with no concept of packets and the DSI hardware does not decode the packet headers and ECC/CS computation is not performed.

  In order to send the RAW data bytes (debug / diagnostic workaround mode):

  - Program the number of bytes to be transferred to the DSI_RAW_DATA_BYTE_COUNT register. The byte count programmed is equal to the total number of raw bytes (not to be interpreted as the byte count of a packet).
  - Write raw data bytes to the DSI Host.
  - Enable RAW_DATA in the DSI_HOST_DSI_CONTROL_0 register.

26.5.4 High-Speed Clock Configurations

In High Speed mode, the Host provides a low-swing, differential DDR clock signal for high-speed data transfers. The high-speed clock lane is configurable via the DSI_HS_CLK_CTRL] field in the DSI_DSI_CONTROL register.

- Free Running Mode /Continuous

The D-Phy clock lane operates in HS free running or continuous mode; i.e., used in systems where the D-Phy clock lane acts as a clock source to the display devices eliminating the need for an alternate oscillator clock source.

- TX Mode/Discontinuous Mode

The D-Phy clock lane operates in burst mode or discontinuous mode; i.e., the D-Phy clock lane remains active only during the HS transmission and stops the clock lane when there are no high-speed transmissions active in any of the data lanes.

Notes: *The Host programs the total horizontal blank period to meet the following criteria to provide enough time for HS <-> LP transitions.*
*(HBLANK) >= (number_of_lanes *(HS_Trail + HS-Exit + CLK_POST + CLK_TRAIL + HS_EXIT + TLPX + TLPX + TCLK_PREPARE + TCLK_ZERO + TCLK_PRE + 1 + TLPX + THS_PREPARE + TDAT_ZERO + 1))*
The parameters in this equation are defined in the DSI_DSI_PHY_TIMING registers.

- *The DSI_HS_CLK_CTRL field in the DSI_DSI_CONTROL_0 register control field for the HS clock lane is programmed when the DSI_HIGH_SPEED_TRANS control field in the DSI_DSI_CONTROL_0 register is programmed for HS packet transmission of packets.*

26.5.4.1 Switch Clock Lane between Continuous and Discontinuous Mode

The clock lane can be switched between continuous and dis-continuous mode DSI_DSI_CONTROL_0[DSI_HS_CLK_CTRL]

- 0: Continuous mode
- 1: Discontinuous mode (HS clock is only active during HS transmissions)

The following programming sequence need to be followed while switching the HS clock lane state:

- Wait for completion of the current transfers/transactions in the DSI and the idle status bit set.
- Update/switch the DSI_HS_CLK_CTRL field in the DSI configuration register DSI_DSI_CONTROL_0.

Note: *The clock lane shall be switched only when there are no high-speed transmissions active on any data lane.*

26.5.5 Ultra-Low Power Sequence

In this procedure, starting from the Stop state, the transmit side drives the TX-ULPS-Rqst state (LP-10) and then drives the TX-ULPS State (LP-00). After this, the clock lane enters the Ultra-Low Power state. If an error occurs, and an LP-01 or LP-11 is detected immediately after the TX-ULPS-Rqst state, the Ultra-Low Power state entry procedure is aborted, and the receive side waits for or returns to the Stop state, respectively.

The Clock lane and Data lanes can be configured to enter and exit ULPS independently/simultaneously:

- Independent control of the ULPS

  The Clock needs to be in the HS state, sending the active clock when data lanes are entering ULPS, When entering ULPS, the data lanes enter the ULPS first, followed by the clock lanes. Similarly, when exiting ULPS, the Data lanes exit the ULPS first, followed by the clock lanes.

  DSI_ULTRA_LOW_POWER_CONTROL register configures the control to enter the ULPS on particular clock/Data lane.

  ULPS Entry

  - Program the data lane to enter ULP sequence, wait for syncpt.
  - If the DSI_HIGH_SPEED_TRANS bit in the DSI_HOST_DSI_CONTROL_0 register is not equal to 0):

    Program the DSI_HIGH_SPEED_TRANS bit to a value less than or equal to 0.
  - Program the clock lane to enter ULP sequence and wait for syncpt

  ULPS Exit

  - Program the clock lanes to exit the sequence, wait for syncpt
  - Program the data lanes to exit the sequence, wait for syncpt

- Simultaneous control of the ULPS

  The DSI_ULTRA_LOW_POWER field in the HOST_DSI_CONTROL register is configured to control the ULPS for all the lanes, i.e., clock and data lanes 0-3 simultaneously.

  ULPS Entry:

  - If the DSI_HIGH_SPEED_TRANS bit in the DSI_HOST_DSI_CONTROL_0 register is not equal to 0:
    - Program DSI_HIGH_SPEED_TRANS to be less than or equal to 0.

- Program the DSI_ULTRA_LOW_POWER field in the HOST_DSI_CONTROL register to be less than or equal to ENTER_ULPM. The DSI_ULTRA_LOW_POWER field controls the ULPS for all lanes (clock and data lanes 0-3) simultaneously.

ULPS Exit:

- Program the DSI_ULTRA_LOW_POWER field in the HOST_DSI_CONTROL register to be less than or equal to EXIT_ULPM.

Notes: *The DSI_TWAKEUP field in the DSI_PHY_TIMING_2 register defines the length of the exit interval in multiples of 512 byte clocks common for all lanes (Clock/Data).*

- *The ULPM entry sequence always starts from the NORMAL state. Software should place the control fields in the NORMAL state after the exit sequence after the last operation is done. That is, in case of a ULPM sequence on clock, data lanes, you are recommended to program the register to the NORMAL state after the data lane exit sequence.*

26.5.6 End of Transmission Packet (EOTp)

The DSI specification defines a dedicated End of Transmission packet (EoTp) at the protocol layer to indicate the end of HS transmission. For backwards compatibility with earlier DSI systems, this EoTp can be enabled or disabled.

For host transactions during the video mode, EOTp should not be programmed in the Packet Sequence register. Software shall program the EOTP packet as part of the host data during host transactions during video mode.

26.5.7 Host Command Packet During Video Mode Transmission

DSI supports transmission of host command packets during vertical blanking in Video mode.

Host packet transmission during vertical blank time is possible with programmable control through the DSI_VID_MODE_CONTROL register to enable and select the line type to trigger the host command packets during video mode. The 0/1/2/4 line types with single short packet transmission as part of video raster can enable simultaneous host transfer in Video mode.

Video blank line types are:

- Line Type = 0 - Blank line starting with VS
- Line Type = 1 - Blank line starting with VE
- Line Type = 2 - Blank line starting with HS
- Line Type = 4 - First Blank line after active starting with HS.

The programming sequence to enable host transactions during Video mode is:

1. Choose the line type 0/1/2/4 (Video Blanking) for simultaneous host transfers during Video mode. Program the DSI_LINE_TYPE field in the DSI_DSI_VID_MODE_CONTROL_0 register to FOUR.
2. Write host data transactions to the host data buffer via the DSI_DSI_WR_DATA_0 register.
3. Simultaneous host transfers are enabled during Video mode via the Video mode control field. Program the DSI_CMD_PKT_VID_ENABLE field in the DSI_DSI_VID_MODE_CONTROL_0 to 1’b1.
4. Wait for FRAME_DONE syncpt from the display controller.
5. Set the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b1.
6. Clear the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b0.

Notes:

- The host packet data transmission cannot be enabled in Video burst mode, where NULL_PACKETs are included with the HSYNC packet.

- Checksum/ECC computation for the host packets can be performed by either the hardware or software by programming the CS/ECC_Enable fields in the DSI_HOST_DSI_CONTROL_0 register.
- After a single VSYNC/VSYNCEND/HSYNC short packet is transmitted, if the Host transmission is enabled, software-constructed Host data packets are transmitted.
- The length of the Host FIFO data is limited to 64 words deep.
- Host data should be pushed into the Host FIFO before the frame is triggered.
- Though the Host FIFO can accommodate a 256-byte packet, maxHost transfer size should not exceed as given in the below expression (units in byte clock count):

  (host_pkt_data_size < HOST_FIFO_size) && (host_pkt_data_size < (line_time – (sol_delay + min HS-LP-HS blanking time needed + 10)))
- Host packets must be aligned to a word boundary. For host packets ending on an unaligned word boundary, there is support for a maximum of 1 such consecutive packets transferring.

  For Unaligned word boundaries, if the end of a packet does not fall on a 32-bit word boundary then the header for the next packet should be posted to the next location of the FIFO. Refer to Figure 110.

  In the worst-case configuration scenario, the number of lanes is 4, and the Host packets end on Byte 0 (that is, a single byte is written in the last word).

26.5.8 Ganged Mode (Odd-Even/Left-Right) Programming

The Tegra X1 DSI supports Ganged mode. This section assumes that the reader is familiar with the DSI specification and normal video mode programing guidelines.

Odd-Even

For Odd-Even Ganged mode, the programming guidelines are based on these assumptions/limitations:

1. Both partitions have a single PLLD and clock source (DSI instances)
2. For VNB, all data lanes end in the same clock cycle aligned to the number of configured lanes.
3. The number of pixels in the 18 bpp packed case is always a multiple of four.
4. The number of active pixels from the display controller is not greater than 4096 pixels.

Programming guidelines/sequence and equations for Odd-Even Ganged mode are listed below:

1. Required inputs are: image resolution, data format, and optional number of lanes
2. Compute the pixel clock frequency from the given resolution:

   pixel frequency = (TotalHorizontalPeriod * TotalVerticalPeriod * FrameRate) / 10^6
3. Determine the shift clock divider value and ganged mode imagesplit information from the number of pixels, data format, and pixel frequency:

   pixel frequency = DSI clock (byte frequency) * 4/clock divider
4. Symmetrical split is possible for all resolutions. Determine the possibility of an asymmetrical split.

   Let n1 = number of lanes for the first partition, and n2 = number of lanes for the second partition from step 3.

   totalNumLanesInGangedMode = n1 + n2
5. Compute the correction pixels needed to align the totalNumPixels, totalNumBytes with totalNumLanesInGangedMode. From Limitation #2 and #3:
   - The pixelsNeededToInitiallyAlign should be such that updated totalNumPixels is a multiple of LCM (total Num Lane sInGangedMode,X)
   - totalNumBytes = totalNumPixels*bytesPerPixel is a multiple of totalNumLanesInGangedMode.

Notes:

- *LCM – least common multiple (X=4 for 18bpp packed, otherwise X=1)*
  If remainder is non-zero/ganged alignment and was not successful (from Step 9), add 'n' number of additional active pixels (zeros/random data) so that the updated image active width is divisible by the LCM above. This is needed to support image split as determined in (Step 4).
- *Add the additional correction/dummy pixels to HFP, determine the new pixel frequency, and compute the byte frequency using the clock divider. Use this information to program the PLLD and shift clock divider. It is okay to add dummy active pixels since panels provide an option using a register field that can be configured to drop/discard the padded pixels.*
- *Update the total number of pixels now with the newly added active pixels to ensure the total number of active pixels does not exceed 4096 pixels. In case the total number of pixels exceeds 4096 pixels, then the frame is considered to not support asymmetrically split for the given configuration (i.e., frame rate and split configurations).*
- *In the 18 bpp packed format case, you cannot achieve alignment over a definite set of iterations for a given frame rate. If the alignment is not achieved within 50-60 pixels, the pixel alignment is assumed to not be possible for the frame for symmetric split at that frame rate.*

6. Determine if the (first partition width) : (second partition width) ratio = n1/n2. If successful, proceed to the next step. If not, ganged width alignment is not successful. Repeat from step 5 by adding more active pixels that still meet the requirement mentioned there.

   splitFactor (for first partition)= n1/ totalNumLanesInGangedMode;

   First Partition Width (active pixels)= CEIL(spliFactor*image width)

   Pixels per line of first Partition = CEIL(splitFactor*totalNumPixels)

   second partition active width = imagewidth – first partition width

   pixels per line of second partition = totalNumPixels - Pixels per line of first Partition

Notes:

- *In consideration of constraint #4, software will ensure that the total number of pixels does not exceed 4096 pixels. If the total number of pixels exceeds 2570 pixels, then the frame does not support asymmetrically split for the given configuration, i.e., frame rate and split configurations.*
- *The number of active pixels added is limited to no more than 60 pixels. If this value exceeds 60 in these iterations, the input image cannot be split as per the split ratio (cannot deviate much from the selected frame rate).*

7. Determine the new pixel frequency (new active pixels and HFP correction) and compute the byte frequency using the clock divider. Use this information to configure the clock parameters for PLLD clock generation and the shift clock divider.
8. Determine the factors of the first partition width and second partition width.

   Select 'x' from the factors of the first partition and 'y' from the factors of the second partition such that x:y = n1/n2. The minimum of 1:1 (symmetrical) is possible for any image.

   - The first group of pixels: ganged start pointer = 0, valid width = x and low width = y
   - The second group: ganged start pointer = x, valid width = y, low width = x
9. In Ganged mode, the packet sequence programming includes HSYNC, HACT, and HFP (total Horizontal blanking period) packets.

   HACT payload size = Total Actual Image Bytes on Data lanes

Initial HFP payload size = totalNumBytesForTheSelectedPartition - numActBytesToBeSent - 16 (16 is because of 4 Bytes - HSYNC, (4+2) Bytes - HACT, (4+2) Bytes - HFP)

10. Compute the correction bytes (per line of partition) to align with the line width and number of lanes (per partition), limitation #2

    correctionBytesPerLane = CEIL(totalNoOfPixelsPerHorzLine *(FDSI/ Fpixel) – (BPP/ Gangedmode_lanes))

    totalNumBytesForTheSelectedPartition = (correctionBytesPerLane * numOfLanesForSelectedPartition)

    total Horizontal Blank(Bytes) = totalNumBytesForTheSelectedPartition - 16 + correctionBytesForAlignment

11. Ganged mode register programming

    - Even Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = 0

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Even partition active width

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Even partition inactive/low width.

    - Odd Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = High width of Even partition

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = odd partition active width.

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = odd partition inactive/low width

12. SOL delay calculation:

    SolFactor = ((Sol2VldDly + 6) * (FDSI / Fpixe)) + 6;

    sol_delay = Ceil (TotalHorzPixelWidth * (FDSI / Fpixel)) -

    Ceil (((SplitFactor * TotalHorzPixelWidth) * BPP) / Gangedmode_lanes) + SolFactor;

    Final_sol_delay = (dcs_mode) ? (sol_delay + 20) : sol_delay

    DSI_DSI_SOL_DELAY_0 = Final_sol_delay

Notes:

- *Sol2VldDly = Hsync start to Pixel valid delay*
- *totalNoOfPixelsPerHorzLine = Total Number of horizontal pixel count.*
- *SplitFactor = DSI_partition_lanes / Gangedmode_lanes.*
- *Gangedmode_lanes = Total number of lanes required in Ganged mode transfer*
- *DSI_partition_lanes = Total lanes in current partition/channel.*
- *BPP = Bytes per pixel*

Left-Right:

For Left-Right Ganged mode, the programming guidelines are based on these assumptions/limitations:

- Both partitions have a single PLLD and clock source (DSI instances)
- For VNB, all data lanes end in the same clock cycle aligned to the number of configured lanes (per DSI specification).
- The number of pixels in the 18 bpp packed case is always aligned to 4 pixels (per DSI specification).
- The number of active pixels from the display controller is not greater than 4096 pixels.

- For timing constraints that are applicable to overlap/dummy pixels, refer to the Overlap Pixels Packet Sequence subsection.

Programming guidelines/sequence and equations for Left-Right Ganged mode are listed below:

1. Required inputs are Image resolution,data format, and optional number of lanes
2. Compute the pixel clock frequency from the given resolution
   pixel frequency = (TotalHorizontalPeriod * TotalVerticalPeriod * FrameRate) / 10^6
3. Determine the possible shiftclock divider value and ganged mode image split the information from the number of pixels, data format, and pixel frequency.

   pixel frequency = DSI clock (byte frequency * 4/clock divider)
4. Symmetrical split is possible for all resolutions. Determine the possibility of the asymmetrical split.
5. Let n1=number of lanes for first partition and n2 = number of lanes for second partition from step 3

   Total number of DSI data lanes required for video refresh:
   totalNumLanesInGangedMode = n1 + n2
6. Compute the correction/additional pixels needed to align the total number of pixels and total number of bytes with totalNumLanesInGangedMode.

   From Limitation #3:

   - The pixelsNeededToInitiallyAlign should be such that an updated totalNumPixels is a multiple of LCM (totalNumLanesInGangedMode,X)
   - totalNumBytes = totalNumPixels*bytesPerPixel
     Ensure the total number of bytes is a multiple of totalNumLanesInGangedMode

   ---

   Note: *For the least common multiple (LCM), X = 4 for 18 bpp packed; otherwise X = 1*

   In the 18 bpp packed format, in case the alignment does not complete over a definite set of iterations for a given frame rate, check that the total additional pixels added to the align is not greater than 50-60 pixels. If this range is exceeded, the pixel alignment is not possible, and the frame is not considered to be asymmetrically split for the 18 bpp packed case (for that frame rate).

   Add the additional correction pixels to the HFP, determine a new pixel frequency, and compute the byte frequency using the clock divider. Use this information to configure the clock parameters for PLLD clock generation and the shift clock divider.
7. Determine the number of pixels per split on updatedPixelCount from step 6:

   - Determine the first partition width, and then the remaining pixels will be in the second partition.

     splitFactor (for first partition)= n1/ totalNumLanesInGangedMode

     First Partition Width (active pixels)= CEIL(splitFactor*image width)

     Pixels per line of the first partition = CEIL(splitFactor*totalNumPixels)

     second partition active width = imagewidth – first partition width

     pixels per line of the second partition = totalNumPixels - Pixels per line of the first partition.

   Note: *For 18 bpp packed, software should recheck if each of the widths determined above is multiple of 4. Otherwise, adjust the active pixels of the partition (increase) to align to 4 pixels and correspondingly decrease the pixels in the HFP (to still meet the line time) accordingly. Adjust pixels in the second partition as well.*

   ---

   numActBytesToBeSent = partition active width * bytesPerPixel

   totalNumBytesForTheSelectedPartition = pixels per line of selected partition * bytesPerPixel

8. In Ganged mode, the packet sequence programming includes HSYNC, HACT, and HFP (total horizontal blanking period) packets.

   HACT payload size = Total Actual Image Bytes on Data lanes

   Initial HFP payload size = totalNumBytesForTheSelectedPartition - numActBytesToBeSent - 16

   (16 is because of 4 bytes - HSYNC, (4+2) Bytes - HACT, (4+2) Bytes - HFP)

9. Compute the correction bytes (per line of partition) to align with line width and number of lanes (per partition) (limitation #2)

   Correction Bytes added per lane (correctionBytesPerLane) = CEIL(totalNoOfPixelsPerHorzLine *(FDSI/ Fpixel) – (BPP/ Gangedmode_lanes))

   Correction Bytes added per partition (totalNumBytesForTheSelectedPartition) = correctionBytesPerLane * numOfLanesForSelectedPartition

   total Horizontal Blank(Bytes) = totalNumBytesForTheSelectedPartition – 16 + correctionBytesForAlignment

10. Ganged mode register programming

    - Left Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = 0;

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Total horizontal active width

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Total horizontal width – Total horizontal active width.

    - Right Partition Programming

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = leftPartitionWidth (for right partition)

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Total horizontal active width

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Total horizontal width – Total horizontal active width.

11. SOL delay:

    SolFactor = ((Sol2VldDly + 8) * (FDSI / Fpixe)) + 6;

    sol_delay = Ceil (TotalHorzPixelWidth * (FDSI / Fpixel)) -

    Ceil (((SplitFactor * TotalHorzPixelWidth) * BPP) / Gangedmode_lanes) + SolFactor;

    Final_sol_delay = (dcs_mode) ? (sol_delay + 20) : sol_delay

    DSI_DSI_SOL_DELAY_0 = Final_sol_delay

Notes:

- *Sol2VldDly = Hsync start to Pixel valid delay*
- *totalNoOfPixelsPerHorzLine = Total Number of horizontal pixel count.*
- *SplitFactor = DSI_partition_lanes / Gangedmode_lanes.*
- *Gangedmode_lanes = Total number of lanes required in Ganged mode transfer*
- *DSI_partition_lanes = Total lanes in current partition/channel.*
- *BPP = Bytes per pixel*

Recommended Packet Sequence

Table 145: Payload Size Table

Packet	Payload Size (Bytes)
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 16

1. B = 2, 2.25 or 3, depending on pixel format.

Table 146: Line Type Packet Sequences

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	RGB	3	BLNK	4				
4	HS	0								
5	HS	0	RGB	3	BLNK	4				

Alternate Packet Sequence

Table 147: Payload Size Table

Packet	Payload Size (Bytes)
HBP	(T_{HFP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

1. B = 2, 2.25 or 3, depending on pixel format.

Table 148: Line Type Packet Sequences - Non-Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	BLNK	2	RGB	3	BLNK	4		
4	HS	0								
5	HS	0	BLNK	2	RGB	3	BLNK	4		

Overlap Pixels Packet Sequence

The reader is expected to be familiar with the Ganged Mode, Left-Right Mode, DSI specification, and normal video mode programming guidelines before reading this subsection.

Left Frame Section (DSI0):

DSI Packet sequence programming for dummy/overlap pixels

The additional dummy, overlap pixels shall utilize the horizontal blanking interval to mitigate the extra line time required for pixel padding.

Recommended Packet Sequence

Table 149: Payload Size Table

Packet	Payload Size (Bytes)
ACTIVE	(T_{ACTIVE} + Tlp_do + Tlp_op) * B

Table 149: Payload Size Table

Packet	Payload Size (Bytes)
HFP	((TBlank – Tlp_do - Tlp_op) * B) – 16

- B = 2, 2.25 or 3, depending on pixel format.
- Tlp_do = Left partition dummy pixel count

  i.e., DSI_DUMMY_PIX_CNT_0[DSI_DUMMY_PIX_CNT_0_LEFT_DUMMY_PIX_CNT] + 1
- Tlp_op = Left partition overlap pixel count
  - DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1'b1
  - DSI_GANGED_START_POINTER = 13'b0
  - DSI_GANGED_VALID_HIGH_WIDTH = TActive(LeftFrameSection) + Tlp_op
  - DSI_GANGED_VALID_LOW_WIDTH = TActive(H-Total) - DSI_GANGED_VALID_HIGH_WIDTH

  Where: TActive(H-Total) = TActive(LeftFrameSection) + TActive(RightFrameSection)
  - Horizontal Blanking (effective) can be determined in two ways based on the choice of blanking interval selection in the video raster.

    (THBP - Tlp_do) + (THFP – Tlp_op)

Table 150: Line Type Packet Sequences

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	RGB	3	BLNK	4				
4	HS	0								
5	HS	0	RGB	3	BLNK	4				

Notes:

- *The dummy, overlap lap feature is applicable only for Ganged Mode Left-Right mode configuration.*
- *Dummy, Overlap pixels padded shall comply VESA timing for horizontal blanking as long as panel vendor/display driver ensure proper raster.*
- *The resultant horizontal blanking time with padded dummy, overlap pixels defined in the table shall be positive.*

Alternate Packet Sequence

Table 151: Payload Size Table

Packet	Payload Size (Bytes)
HBP	(T_{HBP} –Tlp_do) * B – 14
ACTIVE	(T_{ACTIVE} + Tlp_do + Tlp_op) * B
HFP	(T_{HFP} – Tlp_op) * B – 8

- B = 2, 2.25 or 3, depending on pixel format.
- Tlp_do = Left partition dummy pixel count

  i.e., DSI_DUMMY_PIX_CNT_0[DSI_DUMMY_PIX_CNT_0_LEFT_DUMMY_PIX_CNT] + 1

- Tlp_op = Left partition overlap pixel count
  - DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1'b1
  - DSI_GANGED_START_POINTER = 13'b0
  - DSI_GANGED_VALID_HIGH_WIDTH = TActive(LeftFrameSection) + Tlp_op
  - DSI_GANGED_VALID_LOW_WIDTH = TActive(H-Total) - DSI_GANGED_VALID_HIGH_WIDTH

  Where: TActive(H-Total) = TActive(LeftFrameSection) + TActive(RightFrameSection)
  - Horizontal Blanking (effective) can be determined in two ways based on the choice of blanking interval selection in the video raster.

    (THBP + THFP) – (Tlp_do + Tlp_op)

Table 152: Line Type Packet Sequences - Non-Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	BLNK	2	RGB	3	BLNK	4		
4	HS	0								
5	HS	0	BLNK	2	RGB	3	BLNK	4		

Notes:

- *The dummy, overlap lap feature is applicable only for Ganged Mode Left-Right mode configuration.*
- *Dummy, Overlap pixels padded shall comply VESA timing for horizontal blanking as long as panel vendor/display driver ensure proper raster.*
- *The resultant horizontal blanking time with padded dummy, overlap pixels defined in the table shall be positive.*

Right Frame Section (DSI1):

DSI Packet sequence programming for dummy/overlap pixels

The additional dummy and Overlap pixels shall utilize the horizontal blanking interval to mitigate the extra line time required for pixel padding.

Recommended Packet Sequence

Table 153: Payload Size Table

Packet	Payload Size (Bytes)
ACTIVE	(T_{ACTIVE} + Trp_do + Trp_op) * B
HFP	((TBlank – Trp_do - Trp_op) * B) – 16

- B = 2, 2.25 or 3, depending on pixel format.
- Trp_do = Right partition dummy pixel count

  i.e., DSI_DUMMY_PIX_CNT_0[DSI_DUMMY_PIX_CNT_0_RIGHT_DUMMY_PIX_CNT] + 1
- Trp_op = Right partition overlap pixel count
  - DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1'b1
  - DSI_GANGED_START_POINTER = (Start pointer of RightFrameSection) - Trp_op

- DSI_GANGED_VALID_HIGH_WIDTH = TActive(RightFrameSection) + Trp_op
- DSI_GANGED_VALID_LOW_WIDTH = TActive(H-Total) - DSI_GANGED_VALID_HIGH_WIDTH

Where: TActive(H-Total) = TActive(RightFrameSection) + TActive(RightFrameSection)

- Horizontal Blanking (effective) can be determined in two ways based on the choice of blanking interval selection in the video raster.

  (THBP + THFP) – (Trp_do + Trp_op)

Table 154: Line Type Packet Sequences

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	RGB	3	BLNK	4				
4	HS	0								
5	HS	0	RGB	3	BLNK	4				

Notes:

- *The dummy, overlap lap feature is applicable only for Ganged Mode Left-Right mode configuration.*
- *Dummy and Overlap pixels padded shall comply VESA timing for horizontal blanking as long as panel vendor/display driver ensure proper raster.*
- *The resultant horizontal blanking time with padded dummy, overlap pixels defined in the table shall be positive.*

Alternate Packet Sequence

Table 155: Payload Size Table

Packet	Payload Size (Bytes)
HBP	(T_{HBP} –Trp_do) * B – 14
ACTIVE	(T_{ACTIVE} + Trp_do + Trp_op) * B
HFP	(T_{HFP} – Trp_op) * B – 8

- B = 2, 2.25 or 3, depending on pixel format.
- Trp_do = Left partition dummy pixel count

  i.e., DSI_DUMMY_PIX_CNT_0[DSI_DUMMY_PIX_CNT_0_RIGHT_DUMMY_PIX_CNT] + 1
- Trp_op = Left partition overlap pixel count
  - DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1’b1
  - DSI_GANGED_START_POINTER = (Start pointer of RightFrameSection) - Trp_op
  - DSI_GANGED_VALID_HIGH_WIDTH = TActive(RightFrameSection) + Trp_op
  - DSI_GANGED_VALID_LOW_WIDTH = TActive(H-Total) - DSI_GANGED_VALID_HIGH_WIDTH

  Where: TActive(H-Total) = TActive(LeftFrameSection) + TActive(RightFrameSection)
  - Horizontal Blanking (effective) can be determined in two ways based on the choice of blanking interval selection in the video raster.

    (THBP - Trp_do) + (THFP – Trp_op)

Table 156: Line Type Packet Sequences - Non-Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	BLNK	2	RGB	3	BLNK	4		
4	HS	0								
5	HS	0	BLNK	2	RGB	3	BLNK	4		

Notes:

- *The dummy, overlap lap feature is applicable only for Ganged Mode Left-Right mode configurations.*
- *Dummy, Overlap pixels padded shall comply VESA timing for horizontal blanking as long as panel vendor/display driver ensure proper raster.*
- *The resultant horizontal blanking time with padded dummy, overlap pixels defined in the table shall be positive.*

26.5.9 Overlay Pixels Programming Guidelines

Overlay pixels refer to the additional number of active pixels sent per partition to meet the panel requirements with original line timing maintained.

Left partition:

- DSI_GANGED_START_POINTER = 0
- DSI_GANGED_VALID_HIGH_WIDTH = Actual High Width + overlay pixels count
- DSI_GANGED_VALID_LOW_WIDTH = Total width - DSI_GANGED_VALID_HIGH_WIDTH
- Hblank time adjusted per active line (since active time increased, blanking time reduced to maintain same line timing)
- Restriction on number of overlay pixels that can be added.
- No change in sol_delay
- For 18bpp packed case, even overlay pixels are consider to make sure total number of pixels is multiple of 4, alternatively, overlay pixels%4 == 0

Right partition:

- DSI_GANGED_START_POINTER = actual start pointer – overlay pixels count
- DSI_GANGED_VALID_HIGH_WIDTH = Actual High Width + overlay pixels count
- DSI_GANGED_VALID_LOW_WIDTH = Total width - DSI_GANGED_VALID_HIGH_WIDTH
- Hblank time adjusted per active line (since active time increased, blanking time reduced to maintain same line timing)
- Restriction on number of overlay pixels that can be added.
- No Change in sol_delay
- For 18bpp packed case, even overlay pixels are consider to make sure total number of pixels is multiple of 4, alternatively, overlay pixels%4 == 0

In asymmetric split cases, the number of overlay pixels is the same across left and right partitions and not in the ratio of asymmetric split ratio.

Example 1: (1080x1680 2x4 24 bpp)

- Image resolution: 1080x1680 @ 59.8fps (CVT reduced blanking)
- Data format: 24bpp
- Htotal = 1080+160 = 1240
- Vtotal = 1680+46 = 1726
- Pixel freq = 128MHz
- Line time = 9.6875us
- Split ratio: 2x4 (Symmetrical case)
- Split type: Left/right
- Video mode: Non burst video
- Actual left partition active width = 540, pixel index 0-539
  Ganged mode en = 1, start pointer = 0, high width =540, low width = 540
- Actual right partition active width = 540, pixel index 540-1079
  Ganged mode en = 1, start pointer =540, high width = 540, low width = 540
- Packet structure: HSYNC (4 bytes) + HACT (4 + 0x654 + 2) + HBLANK (4+0xe0+2) = 1860 bytes = 620 pixels = (540+80)
- Sol_delay: refer to Section 26.5.2.1: SOL Delay Programming.

Example for Overlay pixel count = 8:

- Left partition active width = 548, pixel index 0-547
  Ganged mode en = 1, start pointer = 0, high width =548, low width = 532
- Right partition active width =548, pixel index 532-1079
  Ganged mode en = 1, start pointer =532, high width =548, low width = 532
- Hblank timing = actual hblank timing-active width with overlay pixels timing included
- Packet structure: HSYNC (4 bytes) + HACT (4 + 0x66c + 2) + HBLANK (4+0xc8+2) = 1860 bytes = 620 pixels = (548+72)
- Sol_delay: no change in programming

Example 2:

- Image resolution: 2560x1600 @ 60fps (CVT reduced blanking)
- Data format: 24bpp
- Htotal = 2560+160 = 2720 pixels
- Vtotal = 1600+46 = 1646 lines
- Pixel clock frequency = 268.5MHz
- Line time = 10.13 µs
- Split ratio: 2x4 (Symmetrical case)
- Split type: Left/right
- Actual left partition active width = 1280, pixel index 0-1279
  Ganged mode en = 1, start pointer = 0, high width = 1280, low width = 1280
- Actual right partition active width = 1280, pixel index 1280-2559
  Ganged mode en = 1, start pointer = 1280, high width = 1280, low width = 1280
- Sol_delay: refer to Section 26.5.2.1: SOL Delay Programming.

Example for Overlay pixel count = 10:

- Left partition active width = 1290, pixel index 0-1289
  Ganged mode en = 1, start pointer = 0, high width = 1290, low width = 1270
- Right partition active width = 1290, pixel index 1270-2559
  Ganged mode en = 1, start pointer = 1270, high width = 1290, low width = 1270
- Hblank timing = actual Hblank timing-active width with overlay pixels timing included
- Sol_delay: no change in programming

26.5.10 Soft Reset Programming

Soft reset is asserted in the DSI through the LEG_DSI_ENABLE control field of the DSI_DSI_POWER_CONTROL_0 register. When soft reset is asserted, the DSI controller transitions the internal states to default mode. When it is asserted between valid transactions, the current transaction is halted, the internal states are moved to default mode, and a new transaction starts by programming the desired values following the programming sequence explained in the other sections.

Note: *For Host transmissions enabled through the DSI_HOST_TRIGGER field of the DSI_DSI_TRIGGER_0 register, soft reset cannot be issued between a transaction, and software will write 1'b0 to the register to clear the trigger bit.*

26.5.11 Function Programming

26.5.11.1 ECC Generation

For precise details on the calculation of the ECC field of the packet header, reference should be made to the MIPI Alliance Standard for Display Serial Interface. However, this is an overview of how the ECC can be created quite simply.

Each bit of the ECC byte is the result of XORing a number of bits from the packet header together. Which header bits contribute to which ECC bit is contained in a special table which can be used to calculate the ECC as follows:

```
const UCHAR ecc_parity[24] = { 0x07, 0x0b, 0x0d, 0x0e, 0x13, 0x15, 0x16, 0x19,
                               0x1a, 0x1c, 0x23, 0x25, 0x26, 0x29, 0x2a, 0x2c,
                               0x31, 0x32, 0x34, 0x38, 0x1f, 0x2f, 0x37, 0x3b
                             };
ULONG packet_header;
UCHAR ecc_byte;
UINT  i;

// Assume bottom 24 bits of packet_header
// contains header ID and byte count, then ...

ecc_byte = 0;
for (i = 0; i < 24; i++) {
  ecc_byte ^= ((packet_header >> i) & 1) ? ecc_parity[i] : 0x00;
}
packet_header |= (ULONG)(ecc_byte) << 24;
```

Note: *The table in the DSI specification actually contains 64 entries. Since short packets have been fixed in length to be the same as a long packet header since the original specification was written, there will now always be just 24 data bits in the packet header, so only 24 entries are needed.*

26.5.11.2 CRC Insertion

The DSI checksum used for long packets is derived using the generator polynomial $x^{16}+x^{12}+x^5+x^0$. Details of this can be found in Section 9.6 of the MIPI DSI specification. To understand how the CRC is generated, think of the packet data as consisting of a continuous serialized stream of bits, rather than a sequence of 8-bit bytes. When thought of in this way, it is relatively straight forward to generate the CRC. Reproduced below is an abridged version of the example C code contained in Appendix B of the MIPI DSI specification.

```
// Polynomial, bit reversed form (since DSI transmits LSB first) …
const unsigned short CRC16GenerationCode = 0x8408;

unsigned short CalculateCRC16( unsigned char *DataStream_ptr, unsigned short NumberOfDataBytes)

{
  unsigned short ByteCounter;
  unsigned char  BitCounter;
  unsigned char  CurrentData;
  unsigned short CRC16Result = 0xFFFF;

  if (NumberOfDataBytes > 0) {
    for (ByteCounter = 0; ByteCounter < NumberOfDataBytes; ByteCounteer++) {
      CurrentData = DataStream_ptr[ByteCounter];
      for (BitCounter = 0; BitCounter < 8; BitCounter++) {
        if ((CRC16Result & 0x0001) ^ (CurrentData ^ 0x0001))
          CRC16Result = ((CRC16Result >> 1) & 0x7FFF) ^ CRCGenerationCode;
        else
          CRC16Result =  (CRC16Result >> 1) & 0x7FFF;
        CurrentData = (CurrentData >> 1) & 0x7F;
      }
    }
  }
  return CRC16Result;
}
```

This code may not be very high-performance due to the inner for loop which iterates over bits, rather than bytes. While it is instructive and could form the basis of a reference piece of code, it is not recommended where performance is important. There are many documented methods of performing this calculation in parallel in order to speed up the computations. These methods are beyond the scope of this document.

26.5.12 Read Data Return

DSI is a bidirectional interface. Data is returned from the peripheral display device only after the display controller has requested information by issuing a Bus Turn Around (BTA). All returned data is written into a FIFO that can be read using Host reads of a DSI register.

26.5.12.1 Reading Peripheral Registers

A typical application of a BTA is in the reading of a register from the display peripheral. This is achieved in the following way:

1. Set up the DSI interface to be in Host-driven command mode.
2. Set the DSI_MAX_THRESHOLD to 3.
3. Set the HOST_TX_TRIG_SRC field of the HOST_DSI_CONTROL register to FIFO_LEVEL.
4. Set the PKT_BTA field of the HOST_DSI_CONTROL register to ENABLE.

5. Write a DCS READ command packet (see Section 8.8.8.2 of the MIPI DSI specification) into the Command FIFO by writing to the DSI_WR_DATA register.

This will result in the transmission of a DCS READ packet to the peripheral, the initiation of a BTA and – assuming the peripheral received the packet without error – the return of the requested data to the Host Read Return FIFO. The data can then be read from the FIFO by reading the DSI_RD_DATA register.

26.5.12.2 Bus Turn Around

- BTA is only supported for Host driven Command Mode interface. There will be no BTA during video mode transmission.
- Whether or not a BTA is initiated is controlled by the `PKT_BTA` field of the HOST_DSI_CONTROL register.
- The DSI Read Return FIFO is 4 bytes wide and 8 entries deep, so 32 bytes in total. this is enough to hold 8 short packets, or a mixture of short and long packets. The length of long packets must be severely restricted.
- Hardware does not perform ECC or CS checks on the read return data. Entire packets are simply made available to software to perform whatever checks they desire.
- There is the ability to increment a sync point counter on the arrival of the returned data or on the receipt of an error report in the event there was a problem with the read packet transaction.
- Software will ensure to program the byte clock frequency dsi_clk greater than 52 MHz, when performing BTA transactions.

26.5.12.3 BTA – Response Time Parameter

When a peripheral receives a READ Request, it is expected that a Bus Turn Around will immediately follow to extend support for the display device that cannot handle the request immediately after the Read request is issued.

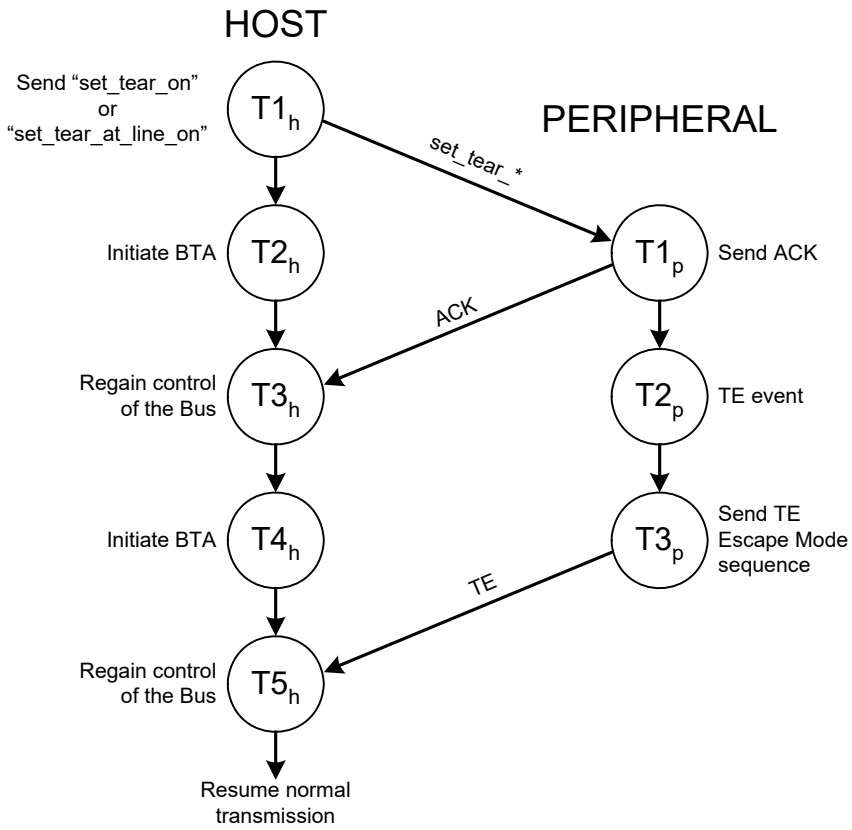
The DSI_DSI_BTA_TIMING_0 register defines DSI_TPKTBTA to allow a programmable delay between the end of host transmission and generation of BTA request for extending support for the display devices with slow response.

Note: *Valid programmable values for all the DSI_DSI_BTA_TIMING_0 register fields (DSI_TTAGO/ DSI_TTASURE/DSI_TTAGET/DSI_TPKTBTA) are 0 to 254. Invalid value is 255.*

26.5.13 Tearing Effect

In order to synchronize the update of a Command Mode display with data from the Host, a signal from the display can be sent to indicate when it is safe to proceed with the transmission of new data. This is the Tearing Effect reporting signal. Refer to Section 8.12 “TE Signaling in DSI” of the DSI specification for more details.

Figure 112: Tearing Effect State Transition Diagram



Programmatically, this is achieved in the following way:

1. Send SET_TEAR_ON or SET_TEAR_AT_LINE_ON command with the PKT_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This is state T1h.
2. Wait for ACK to come back from the peripheral. This is states T2h and T3h.
3. Set the IMM_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This will cause the D-PHY to go into BTA without having to send a command first. This is state T4h.
4. Wait for TE return byte from the peripheral. This is state T5h.

26.5.14 Pad Calibration Programming

- Configure the MIPI calibration clock to operate at 72 MHz via programming the register CLK_RST_CONTROLLER_CLK_SOURCE_UART_FST_MIPI_CAL_0.
- Enable the MIPI calibration clock via Set CLK_ENB_MIPI_CAL to 1.
- Release reset to mipi_cal logic by deasserting SWR_MIPI_CAL_RST to 0
- Configure the DSI pad/Bias pads to appropriate values

  DSI_PAD_CONTROL_3_0= 0x0

  DSI_PAD_CONTROL_4_0 = 0x0

  MIPI_CAL_MIPI_BIAS_PAD_CFG0_0 = 0x0;

  MIPI_CAL_MIPI_BIAS_PAD_CFG1_0 = 0x00020000

  MIPI_CAL_MIPI_BIAS_PAD_CFG2_0 = 0x00000000
- Configure MIPI calibration settings for DSI pads for setup MIPI_CAL_DSI* _MIPI_CAL_CONFIG [*SELDSI*/ *OVERIDEDSI*/ *HSPDOSDSI*/ *HSPUOSDSI*/ *TERMOSDSI*].

  MIPI_CAL_DSI*_MIPI_CAL_CONFIG_0

- Enable all the DSI lanes that require calibration driving LP_11 state.
- MIPI calibration start via - MIPI_CAL_MIPI_CAL_CTRL_0[MIPI_CAL_STARTCAL]
- Monitor the status via MIPI_CAL_CIL_MIPI_CAL_STATUS_0[*].

26.5.15 Error Reporting

There is no actual error recovery circuitry in the hardware, only error reporting. Any error that is reported via a protocol-level packet will be processed like all other return data and will be written to the data return FIFO for processing by the Host / Software.

Low level hardware errors such as bus contention will not be flagged, but will increment counters that can be queried by software so as to determine the reliability of the link.

26.5.15.1 Acknowledge with Error Report

The peripheral device (display) can be instructed to return an error report along with an acknowledge at the end of a transmission sequence. This is achieved by setting the PKT_BTA field in the HOST_DSI_CONTROL register to ENABLE. When this is done, there are several outcomes as shown in the table below.

Table 157: Peripheral Response to Various Conditions

Condition	Non-Read Packet	Read Packet
No error	ACK	Read Data
Corrected single bit error	ACK with ERR	Read Data + ACK with ERR
Non-corrected multi-bit error	ACK with ERR	ACK with ERR
SOT, EOT, VI ID or other D-PHY error	ACK with ERR	ACK with ERR

It is the responsibility of software to read this data from the packet returned in the packet return FIFO and process as required. No action will be taken by the hardware based on the error report returned in the ACK packet.

ACK Return

If a BTA request is enabled and there is no error, then the peripheral will return an ACK trigger to the DSI hardware. This single byte trigger has the value 0x84. Since the return FIFO is 32 bits wide, the 0x84 value will be placed in the bottom 8 bits of the FIFO.

Read Data

If a read command packet is sent, then BTA request should be enabled to allow the peripheral to return the read data. If no error occurs in the transmission of the read packet, or a corrected single bit error occurs, then the read data will be returned to the host DSI hardware in the form of a read packet. The precise form will depend on the type of read packet transmitted. Refer to the MIPI DSI specification Section 8.10 for details.

If a corrected error occurs during the transmission of the read command, the requested read data will be returned in the normal way, but an ACK with Error report packet will be appended to the read return data packet.

ACK with Error Report

The error report is contained in the 16 payload bits of a special short packet returned by the peripheral. The bits allocations of the packet data are detailed in Section 8.9.5 of the MIPI DSI specification but are repeated here for convenience.

Table 158: Error Report Bit Assignments

Bit	Description
0	SOT error
1	SOT sync error
2	EOT sync error
3	Escape Mode Entry Command error

Table 158: Error Report Bit Assignments

Bit	Description
4	Low-Power Transmit Sync Error
5	HS Receive Timeout error
6	False Control Error
7	RESERVED
8	ECC Error, single bit (corrected)
9	ECC Error, multi-bit (not corrected)
10	CS Error (long packet only)
11	DSI data type not recognized
12	DSI Virtual Channel ID invalid
13	RESERVED
14	RESERVED
15	RESERVED

26.5.16 Time Outs

There are three compulsory and one optional time out counters required by the MIPI DSI specification for Processors (display controller). Each timer will consist of a simple counter which will count DSI byte clocks. The counters will reset to 0 on an event and will then simply increment their count every DSI byte clock cycle. If the event that the time out is protecting occurs before the time out reaches its terminal count, the counter will simply stop counting and hold its value. If the counter reaches software programmed maximum count before the expected event occurs, the counter will stop counting and hold its count value. Any action that is required by the MIPI DSI specification upon reaching the time out will also be performed by the hardware.

Table 159: Time-out Counter Summary

Time Out Name	Abbreviation	Start Condition	Length Greater Than	Action
HS Transmit TO	HTX_TO	SOT	Longest HS sequence	EOT + LP-11
LP Receive TO	LRXH_TO	LP Rx start	LTXP_TO (on periph.)	LP-11
Turn Around TO	TA_TO	BTA start	BTA response time	LP-11
Peripheral Reset	PR_TO	Reset Entry CMD	Peripheral response time	None.

In the table, the Peripheral Reset time out is the only timer that is optional. All the other timers are required by the MIPI DSI specification. Note that under “Action”, the listed operations are forced on the interface by the D-PHY under the instruction of the protocol layer (hardware). In the case of the optional Peripheral Reset time out, there is no action since this time out simply exists to issue a reset to the peripheral. Once the reset command is issued, the only further action required is to wait for an appropriate length of time to allow the peripheral to go through its reset sequence.

In the case of the HTX_TO, LRXH_TO and TA_TO time outs, the reaching of a terminal count will not only cause the action as listed in the table below, but will also cause a tally counter to increment so that software can read the register and determine if any of the time outs have occurred. Software will be able to reset these tally counters to 0 by writing to the tally register. The value written will be irrelevant. The PR_TO will report its operation in a different manner. When the PR_TO counter is actually counting, there will be a status bit that reads as ‘0’. When the PR_TO terminal count is reached and time out ends, the status bit will become ‘1’. This will allow software to determine when the peripheral has been reset.

Table 160: Time Out and Tally Registers

Register	Field	Description
DSI_TIMEOUT_0	HTX_TO	High Speed Transmit time out duration
	LRXH_TO	Low Power Receive time out duration
DSI_TIMEOUT_1	TA_TO	Turn Around time out duration
	PR_TO	Peripheral Reset duration

Table 160: Time Out and Tally Registers

Register	Field	Description
DSI_TO_TALLY	HTX_TALLY	High Speed Transmit time out Tally
	LRXH_TALLY	Low Power Receive time out Tally
	TA_TALLY	Turn Around time out Tally
	P_RESET_STATUS	Peripheral Reset Status: 0= Reset, 1 = Ready

26.6 MIPI-DSI Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

26.6.1 DSI_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15 16 = COND_16 17 = COND_17 18 = COND_18 19 = COND_19 20 = COND_20 21 = COND_21 22 = COND_22 23 = COND_23 24 = COND_24 25 = COND_25 26 = COND_26 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31
7:0	0x0	INDX: syncpt index value

26.6.2 DSI_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all affected Host1x clients, then clear all SOFT_RESETs.

26.6.3 DSI_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

26.6.4 DSI_CTXSW_0

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module, and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0xf000f800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

Note: *PACKET DEFINITIONS*
Packet for Display Controller -> DSI communication:
Line Type is used to convey the type of video line from the display controller to the DSI block. The line type implies the generation of one of the associated packet sequences as defined in the DSI packet sequence registers (see below). Used to construct packet headers. All packet headers are now 32-bits wide regardless of whether they are short or long packets. Used for validation infrastructure only.

26.6.5 DSI_DSI_RD_DATA_0

DSI Read Return Data

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_RD_DATA: Each read to this register will pop 32 bits from the 32-bit wide read return data FIFO. The FIFO has NV_DSI_HOST_DATA_RETURN_FIFO_DEPTH entries.

26.6.6 DSI_DSI_WR_DATA_0

Host FIFO Write Input

Offset: 0xa | Byte Offset: 0x28 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_WR_DATA: Each write to this register will push 32 bits into the 32-bit wide Host data FIFO. FIFO has NV_DSI_HOST_DATA_FIFO_DEPTH entries

26.6.7 DSI_DSI_POWER_CONTROL_0

Display Power Control

DSI Enable

Offset: 0xb | Byte Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	LEG_DSI_ENABLE: DSI interface Enable 0 = DISABLE: Disable DSI 1 = ENABLE: Enable DSI

26.6.8 DSI_INT_ENABLE_0

Interrupt Enable Register

Offset: 0xc | Byte Offset: 0x30 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE: interrupt disabled 1 = ENABLE: interrupt enabled

26.6.9 DSI_INT_STATUS_0

Interrupt Status Register

Offset: 0xd | Byte Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

26.6.10 DSI_INT_MASK_0

Interrupt Mask

Setting bits in this register mask the corresponding interrupt but does not clear a pending interrupt and does not prevent a pending interrupt to be generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status to be generated.

Offset: 0xe | Byte Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED: interrupt masked 1 = NOTMASKED: interrupt not masked

26.6.11 DSI_HOST_DSI_CONTROL_0

DSI Control Register When Input is from HOST

Offset: 0xf | Byte Offset: 0x3c | Read/Write: R/W | Reset: 0x00000043 (0bxxxxxxxxxx00x000xx00xx0001000011)

Bit	Reset	Description
21	0x0	FIFO_STAT_RESET: write only bit to clear FIFO underflow/overflow flags. If a new underflow/overflow event occurs during the same time, current access cannot clear the status bits
20	0x0	CRC_RESET: Write only bit. When written with a 1, causes the Verification CRC generator to reset to 0xFFFF_FFFF. If written with 0, it has no effect.
18:16	0x0	DSI_PHY_CLK_DIV: Phy clock divider value for byte clock 0 = DIV1 1 = DIV2
13:12	0x0	HOST_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL: Start of Line signal from the Display Controller. 1 = FIFO_LEVEL: How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE: Determined by a write to the DSI_HOST_TRIGGER field of the DSI_TRIGGER register
9:8	0x0	DSI_ULTRA_LOW_POWER: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
7	0x0	PERIPH_RESET: Initiate an Escape Mode Peripheral Reset. 0 = DISABLE: Causes an Escape Mode Command to be sent to reset the 1 = ENABLE: External display device. Also starts the PR_TO counter. PR_TO state can be checked with the P_RESET_STATUS field in the DSI_TO_TALLY register. Hardware clears this bit upon completion of issuing "Trigger Reset" OR called "Reset Entry" command is sent out.
6	0x1	RAW_DATA: Host raw data mode. In this mode. All data is sent exactly as written. No attempt to decode packet headers is made. This bit will also override the function of the ECC and CS ENABLE fields. No ECC or CS will be generated. This mode is intended as a debug / diagnostic workaround mode only. 0 = DISABLE: Normal mode. 1 = ENABLE: Enable raw data transmission.
5	0x0	DSI_HIGH_SPEED_TRANS: DSI high speed transmission of packets 0 = LOW: Low speed - Note: Unlikely ever to be used. 1 = HIGH: High speed
4	0x0	PKT_WR_FIFO_SEL: Host Write FIFO Select. In video mode, software shall not program PKT_WR_FIFO_SEL=VIDEO. 0 = HOST: Write data to the small host data FIFO only. 1 = VIDEO: Write data to both the host and video line store FIFO, in series. Note: Software shall only program PKT_WR_FIFO_SEL field in Host mode.
3	0x0	IMM_BTA: Generate BTA immediately, e.g., for Tearing Effect reporting. Note: This will generate a BTA and pass control of the D-PHY to the remote peripheral without the need to send any packet. Once the BTA is initiated on interface this bit gets cleared. Later on Host syncpt opdone is returned when the remote peripheral has responded and relinquished control of the bus. 0 = DISABLE: Do not generate BTA 1 = ENABLE: Generate BTA immediately, without waiting for a packet.
2	0x0	PKT_BTA: Generate BTA at the end of Host packets 0 = DISABLE: Do not generate BTA 1 = ENABLE: Generate BTA after the next packet is sent.
1	0x1	CS_ENABLE: enable Hardware Check Sum (CS) for Host packets Note: when CS is disabled, Host is responsible for generating proper CRC and adding the 2-byte CRC to the end of the packet after the payload. 0 = DISABLE: Disable hardware generation of CS (Host must calculate CS). 1 = ENABLE: Enable hardware generation of CS.
0	0x1	ECC_ENABLE: Enable hardware Error Correction Code (ECC) for Host packets Note: when ECC is disabled, Host is responsible for generating proper ECC byte for header 0 = DISABLE: Disable hardware generation of ECC (Host must calculate ECC). 1 = ENABLE: Enable hardware generation of ECC.

26.6.12 DSI_DSI_CONTROL_0

General DSI Control Register

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxx0xx00xx00xx00xx000000)

Bit	Reset	Description
31	0x0	DSI_DBG_ENABLE: Control signal to turn off clock monitoring when enabled for debug, on every DSI byte clock debug signal toggle. Also TX CRC computation aid will turn on.
30	0x0	DFMT_16BPP_SWAP_EN: DSI specification supports different bit ordering (only 16 BPP) in command mode. Command Mode (Default): 16BPP[15:0] -> B[4-0]G[5-0]R[4-0]; Command Mode (SWAP_EN): 16BPP[15:0] -> G[2-0]B[4-0]R[4-0]G[5-3]
20	0x0	DSI_HS_CLK_CTRL: Control for the HS clock lane 0 = CONTINUOUS: HS clock is on all the time. 1 = TX_ONLY: HS clock is only active during HS transmissions.
17:16	0x0	DSI_VIRTUAL_CHANNEL: Virtual channel ID. The virtual channel is sent as part of the packet header and used to distinguish multiple displays.
13:12	0x0	DSI_DATA_FORMAT: Pixel Data format transmitted. Note that although the pixel format is specified in the packet header ID for RGB data packets, this information is ignored by the hardware. Only the information used in this register is used in the construction of RGB data packets. 0 = BIT16P: 16 bpp RGB Packed. 2 bytes used per pixel 1 = BIT18NP: 18 bpp RGB Not-packed. 3 bytes used per pixel 2 = BIT18P: 18 bpp RGB Packed. 2.25 bytes used per pixel 3 = BIT24P: 24 bpp RGB Packed. 3 bytes used per pixel
9:8	0x0	VID_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL: Start of Line signal from the Display Controller. 1 = FIFO_LEVEL: How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE: Determined by a write to the DSI_VID_TRIGGER field of the DSI_TRIGGER register
5:4	0x0	DSI_NUM_DATA_LANES: Number of D-PHY data lanes used by Display for HS transmission. 0 = ONE: 1 data lane. 1 = TWO: 2 data lanes. 2 = THREE: 3 data lanes. 3 = FOUR: 4 data lanes.
3	0x0	VID_DCS_ENABLE: Enable for insertion of DCS commands during Display Controller generated packets. When enabled, the DCS commands defined in the LT3_DCS_CMD and LT5_DCS_CMD fields of the DSI_DCS_CMDS register will be inserted in long packets defined in packet sequence 3 and 5. 0 = DISABLE: No DCS commands will be inserted. 1 = ENABLE: DCS command IDs will be inserted as described above.
2	0x0	DSI_VID_SOURCE: Source of video pixels 0 = DISPLAY_0: Pixels come from "display" 1 = DISPLAY_1: Pixels come from "displayb"
1	0x0	DSI_VID_ENABLE: Video DSI Interface Enable 0 = DISABLE: Disable 1 = ENABLE: Enable
0	0x0	DSI_HOST_ENABLE: Host DSI Interface Enable 0 = DISABLE: Disable 1 = ENABLE: Enable

26.6.13 DSI_DSI_SOL_DELAY_0

Number of Byte-Clock Counts to Wait after Reception

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SOL_DELAY: Start Of Line before generating output packets.

26.6.14 DSI_DSI_MAX_THRESHOLD_0

Maximum Threshold Registers for DSI Related Packets

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	MAX_THRESHOLD: Start draining FIFO once this threshold is met. This register can be used for DBI mode when line packet data exceeds the size of the data FIFO.

26.6.15 DSI_DSI_TRIGGER_0

Manual Transmission Trigger Register

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	DSI_HOST_TRIGGER: A 1 written to this bit will start host transmission when HOST_DSI_CONTROL.HOST_TX_TRIG_SRC is set to IMMEDIATE. Hardware auto clears on operation completion
0	X	DSI_VID_TRIGGER: A 1 written to this bit will start video transmission when DSI_CONTROL.VID_TX_TRIG_SRC is set to IMMEDIATE. Hardware auto clears on operation completion

26.6.16 DSI_DSI_TX_CRC_0

Transmission CRC

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TX_CRC: Long Packet CRC appended to the end of long packets. This CRC is that result of generating a CRC from all transmitted bytes. If DSI_HOST_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted from the Host interface in Command Mode. If DSI_VID_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted during a frame of video when in Video Mode. Note that the hardware will capture the CRC into a separate internal register so that it can continue to calculate the CRC for the next frame without having to wait for software to read the current result.

26.6.17 DSI_DSI_STATUS_0

DSI Status Register

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10	X	DSI_IDLE: Indicated that the DSI is IDLE.
9	X	LB_UNDERFLOW: Indicates that a Line buffer underflow event happened
8	X	LB_OVERFLOW: Indicates that a Line buffer overflow event happened
4:0	X	RD_FIFO_COUNT: Count of how many data words are left in the Host Read Data Return FIFO. Typically, software knows how much data to read from the DSI_RD_DATA register after a Read packet / BTA has been sent / requested, since these transactions are initiated by software. However, under error conditions, insufficient data may have been read from the FIFO. Software should therefore check this field to make sure it is 0 after reading all the information it expected to get.

26.7 Initialization Sequence Registers

26.7.1 DSI_DSI_INIT_SEQ_CONTROL_0

DSI Initialization Sequence Control

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x0000XX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
14:8	X	DSI_FRAME_INIT_BYTE_COUNT: Frame Initialization Sequence Byte Count. This parameter specifies the number of frame initialization sequence cycles to send. If programmed to 0, there is no frame initialization cycle generated. Valid programmable values: 0 to 64. Invalid programmable values: 65 to 127.
0	0x0	DSI_SEND_INIT_SEQUENCE: Send Initialization Sequence (IS) 0 = DISABLE 1 = ENABLE

26.7.2 DSI_DSI_INIT_SEQ_DATA_0_0

DSI Init Sequence Write Data 0

Offset: 0x1b | Byte Offset: 0x6c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_0: DSI Init Sequence Write Data bits 31:0

26.7.3 DSI_DSI_INIT_SEQ_DATA_1_0

DSI Init Sequence Write Data 1

Offset: 0x1c | Byte Offset: 0x70 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_1: DSI Init Sequence Write Data bits 31:0

26.7.4 DSI_DSI_INIT_SEQ_DATA_2_0

DSI Init Sequence Write Data 2

Offset: 0x1d | Byte Offset: 0x74 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_2: DSI Init Sequence Write Data bits 31:0

26.7.5 DSI_DSI_INIT_SEQ_DATA_3_0

DSI Init Sequence Write Data 3

Offset: 0x1e | Byte Offset: 0x78 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_3: DSI Init Sequence Write Data bits 31:0

26.7.6 DSI_DSI_INIT_SEQ_DATA_4_0

DSI Init Sequence Write Data 4

Offset: 0x1f | Byte Offset: 0x7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_4: DSI Init Sequence Write Data bits 31:0

26.7.7 DSI_DSI_INIT_SEQ_DATA_5_0

DSI Init Sequence Write Data 5

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_5: DSI Init Sequence Write Data bits 31:0

26.7.8 DSI_DSI_INIT_SEQ_DATA_6_0

DSI Init Sequence Write Data 6

Offset: 0x21 | Byte Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_6: DSI Init Sequence Write Data bits 31-0

26.7.9 DSI_DSI_INIT_SEQ_DATA_7_0

DSI Init Sequence Write Data 7

Offset: 0x22 | Byte Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_7: DSI Init Sequence Write Data bits 31-0

26.8 Packet Sequence Registers

These registers allow the construction of arbitrary packet sequences associated with various video line types as sent from the Display Controller to the DSI block.

The first digit of the pair of digits in each field below is the sequence number, and the second digit denotes the packet number within each sequence. For example, field PKT_23_ID, defines the ID for packet 3 in sequence 2.

26.8.1 DSI_DSI_PKT_SEQ_0_LO_0

DSI Packet Sequence 0 LO Half

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_0_FORCE_LP: For packet sequence 0, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_02_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_02_ID: Packet 2 Packet ID
22:20	X	PKT_02_SIZE: Packet 2 size pointer
19	X	PKT_01_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_01_ID: Packet 1 Packet ID
12:10	X	PKT_01_SIZE: Packet 1 size pointer
9	X	PKT_00_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_00_ID: Packet 0 Packet ID

Bit	Reset	Description
2:0	X	PKT_00_SIZE: Packet 0 size pointer

26.8.2 DSI_DSI_PKT_SEQ_0_HI_0

DSI Packet Sequence 0 HI Half

Line Type 0 is associated with the first line in the frame and should contain a packet sequence that starts with a VS packet.

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_05_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_05_ID: Packet 5 Packet ID
22:20	X	PKT_05_SIZE: Packet 5 size pointer
19	X	PKT_04_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_04_ID: Packet 4 Packet ID
12:10	X	PKT_04_SIZE: Packet 4 size pointer
9	X	PKT_03_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_03_ID: Packet 3 Packet ID
2:0	X	PKT_03_SIZE: Packet 3 size pointer

26.8.3 DSI_DSI_PKT_SEQ_1_LO_0

DSI Packet Sequence 1 LO Half

Line Type 1 is associated with the last line of Vertical Sync and should contain a packet sequence that starts with a VE packet.

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_1_FORCE_LP: For packet sequence 1, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_12_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_12_ID: Packet 2 Packet ID
22:20	X	PKT_12_SIZE: Packet 2 size pointer
19	X	PKT_11_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_11_ID: Packet 1 Packet ID
12:10	X	PKT_11_SIZE: Packet 1 size pointer
9	X	PKT_10_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_10_ID: Packet 0 Packet ID
2:0	X	PKT_10_SIZE: Packet 0 size pointer

26.8.4 DSI_DSI_PKT_SEQ_1_HI_0

DSI Packet Sequence 1 HI Half

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_15_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_15_ID: Packet 5 Packet ID
22:20	X	PKT_15_SIZE: Packet 5 size pointer
19	X	PKT_14_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_14_ID: Packet 4 Packet ID
12:10	X	PKT_14_SIZE: Packet 4 size pointer
9	X	PKT_13_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_13_ID: Packet 3 Packet ID
2:0	X	PKT_13_SIZE: Packet 3 size pointer

26.8.5 DSI_DSI_PKT_SEQ_2_LO_0

DSI Packet Sequence 2 LO Half

Line Type 2 is associated with any vertical blank line except the first one after active and should contain a packet sequence that starts with an HS packet.

Offset: 0x27 | Byte Offset: 0x9c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_2_FORCE_LP: For packet sequence 2, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_22_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_22_ID: Packet 2 Packet ID
22:20	X	PKT_22_SIZE: Packet 2 size pointer
19	X	PKT_21_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_21_ID: Packet 1 Packet ID
12:10	X	PKT_21_SIZE: Packet 1 size pointer
9	X	PKT_20_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_20_ID: Packet 0 Packet ID
2:0	X	PKT_20_SIZE: Packet 0 size pointer

26.8.6 DSI_DSI_PKT_SEQ_2_HI_0

DSI Packet Sequence 2 HI Half

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_25_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_25_ID: Packet 5 Packet ID
22:20	X	PKT_25_SIZE: Packet 5 size pointer
19	X	PKT_24_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_24_ID: Packet 4 Packet ID
12:10	X	PKT_24_SIZE: Packet 4 size pointer
9	X	PKT_23_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_23_ID: Packet 3 Packet ID
2:0	X	PKT_23_SIZE: Packet 3 size pointer

26.8.7 DSI_DSI_PKT_SEQ_3_LO_0

DSI Packet Sequence 3 LO Half

Line Type 3 is associated with any active line except the first one and should contain a packet sequence that starts with an HS packet and includes an RGB data packet.

Offset: 0x29 | Byte Offset: 0xa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_3_FORCE_LP: For packet sequence 3, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_32_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_32_ID: Packet 2 Packet ID
22:20	X	PKT_32_SIZE: Packet 2 size pointer
19	X	PKT_31_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_31_ID: Packet 1 Packet ID
12:10	X	PKT_31_SIZE: Packet 1 size pointer
9	X	PKT_30_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_30_ID: Packet 0 Packet ID
2:0	X	PKT_30_SIZE: Packet 0 size pointer

26.8.8 DSI_DSI_PKT_SEQ_3_HI_0

DSI Packet Sequence 3 HI Half

Offset: 0x2a | Byte Offset: 0xa8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_35_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_35_ID: Packet 5 Packet ID
22:20	X	PKT_35_SIZE: Packet 5 size pointer
19	X	PKT_34_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_34_ID: Packet 4 Packet ID
12:10	X	PKT_34_SIZE: Packet 4 size pointer
9	X	PKT_33_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_33_ID: Packet 3 Packet ID
2:0	X	PKT_33_SIZE: Packet 3 size pointer

26.8.9 DSI_DSI_PKT_SEQ_4_LO_0

DSI Packet Sequence 4 LO Half

Line Type 4 is associated with the first vertical blanking line after the last active line and should contain a packet sequence that starts with an HS packet. Ordinarily, this packet sequence should be identical to sequence 2.

Offset: 0x2b | Byte Offset: 0xac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_4_FORCE_LP: For packet sequence 4, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_42_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_42_ID: Packet 2 Packet ID
22:20	X	PKT_42_SIZE: Packet 2 size pointer
19	X	PKT_41_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_41_ID: Packet 1 Packet ID
12:10	X	PKT_41_SIZE: Packet 1 size pointer
9	X	PKT_40_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_40_ID: Packet 0 Packet ID
2:0	X	PKT_40_SIZE: Packet 0 size pointer

26.8.10 DSI_DSI_PKT_SEQ_4_HI_0

DSI Packet Sequence 4 HI Half

Offset: 0x2c | Byte Offset: 0xb0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_45_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_45_ID: Packet 5 Packet ID
22:20	X	PKT_45_SIZE: Packet 5 size pointer
19	X	PKT_44_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_44_ID: Packet 4 Packet ID
12:10	X	PKT_44_SIZE: Packet 4 size pointer
9	X	PKT_43_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_43_ID: Packet 3 Packet ID
2:0	X	PKT_43_SIZE: Packet 3 size pointer

26.8.11 DSI_DSI_PKT_SEQ_5_LO_0

DSI Packet Sequence 5 LO Half

Line Type 5 is associated with the first active line. It should contain a packet sequence that starts with an HS packet and includes an RGB data packet. Ordinarily, this packet sequence should be identical to sequence 3.

Offset: 0x2d | Byte Offset: 0xb4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_5_FORCE_LP: For packet sequence 5, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_52_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_52_ID: Packet 2 Packet ID
22:20	X	PKT_52_SIZE: Packet 2 size pointer
19	X	PKT_51_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_51_ID: Packet 1 Packet ID
12:10	X	PKT_51_SIZE: Packet 1 size pointer
9	X	PKT_50_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_50_ID: Packet 0 Packet ID
2:0	X	PKT_50_SIZE: Packet 0 size pointer

26.8.12 DSI_DSI_PKT_SEQ_5_HI_0

DSI Packet Sequence 5 HI Half

Offset: 0x2e | Byte Offset: 0xb8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_55_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_55_ID: Packet 5 Packet ID
22:20	X	PKT_55_SIZE: Packet 5 size pointer
19	X	PKT_54_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_54_ID: Packet 4 Packet ID
12:10	X	PKT_54_SIZE: Packet 4 size pointer
9	X	PKT_53_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_53_ID: Packet 3 Packet ID
2:0	X	PKT_53_SIZE: Packet 3 size pointer

26.9 DCS Command and Packet Length Registers

26.9.1 DSI_DSI_DCS_CMDS_0

DCS command IDs used for Line Types 3 and 5. These command IDs are inserted at the start of the data payload of DCS long packets when the Display Controller is being used to transmit DCS commands.

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:8	X	LT5_DCS_CMD: DCS command for Line Type 5.
7:0	X	LT3_DCS_CMD: DCS command for Line Type 3.

26.9.2 DSI_DSI_PKT_LEN_0_1_0

DSI Packet Lengths 0 and 1

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_1: Packet length 1 (in bytes)
15:0	X	LENGTH_0: Packet length 0 (in bytes)

26.9.3 DSI_DSI_PKT_LEN_2_3_0

DSI Packet Lengths 2 and 3

Offset: 0x35 | Byte Offset: 0xd4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_3: Packet length 3 (in bytes)
15:0	X	LENGTH_2: Packet length 2 (in bytes)

26.9.4 DSI_DSI_PKT_LEN_4_5_0

DSI Packet Lengths 4 and 5

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_5: Packet length 5 (in bytes)
15:0	X	LENGTH_4: Packet length 4 (in bytes)

26.9.5 DSI_DSI_PKT_LEN_6_7_0

DSI Packet Lengths 6 and 7

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_7: Packet length 7 (in bytes)
15:0	X	LENGTH_6: Packet length 6 (in bytes)

26.10 Physical Interface Timing Registers

The DSI PHY timing registers must be initialized before the interface is enabled. Based on the equations provided in the table below, all PHY timing parameters should be programmed in the timing registers. T_{byte} refers to just the DSI controller clock.

Table 161: D-PHY Timing Values

Parameter	MIPI D-PHY Spec Value	Programmed Value
THS-PREPARE	(40 ns + 4*UI , 85 ns + 6*UI)	Min - (40 + (4 T_{bit}))) / T_{byte} Max - ((85 + 6 T_{bit})) / T_{byte}
THS-ZERO	105 ns + 6*UI min.	(105 + (6 T_{bit})) / T_{byte} -2
THS-TRAIL	max(8*UI, 60 ns + 4*UI)	3 + max(8 T_{bit}, 60 + 4 T_{bit}) / T_{byte}
THS-EXIT	100 ns min.	100 / T_{byte}
TLPX	50 ns min.	50 / T_{byte}
TCLK-ZERO	300 ns along with TCLK-PREPARE	260/ T_{byte}
TCLK-PREPARE	(38 ns, 95 ns)	THS-PREPARE timing should guarantee to meet both these timing requirements
TCLK-POST	60 ns + 52*UI min.	(60 + (52 T_{bit})) / T_{byte} - 2
TCLK-TRAIL	60 ns min.	60 / T_{byte}
TCLK-PRE	8 UI	8 T_{bit}
TTA-GO	4 * TLPX	3 * TLPX
TTA-SURE	(TLPX, 2 * TLPX)	TLPX - 1
TTA-GET	5 * TLPX	4 * TLPX
TWAKEUP	1 ms	1 ms/10 ns x 512 {in terms of 512 byte clocks}

Notes:

- T_{byte} = Period of one byte clock in nanoseconds.
- T_{bit} = Period of one bit time in nanoseconds. (T_{bit} = T_{byte} / 8)
- All figures and time periods are in nanoseconds.
- All timing register values are multiples of byte clock period, T_{byte}.
- All fractional results are truncated. No rounding.

26.10.1 DSI_DSI_PHY_TIMING_0_0

DSI D-PHY Timing Register 0

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_THSDEXIT: Time to drive LP11 after HS **Note:** Valid programmable values for all the fields in this register (i.e., DSI_THSPREPR/ DSI_TDATZERO/DSI_THSTRAIL/DSI_THSDEXIT) are 0 to 254. Invalid value is 255.
23:16	X	DSI_THSTRAIL: Time to drive HS flipped bit at EOT
15:8	X	DSI_TDATZERO: Time to drive HS0 before SOT
7:0	X	DSI_THSPREPR: Time to drive LP00 before HS data

26.10.2 DSI_DSI_PHY_TIMING_1_0

DSI D-PHY Timing Register 1

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_TCLKTRAIL: Time to drive HS0 before the clock goes to LP1 **Note:** Valid programmable values for all the fields in this register (i.e., DSI_TTLPX/ DSI_TCLKZERO/DSI_TCLKPOST/ DSI_TCLKTRAIL) are 0 to 254 Invalid value is 255.
23:16	X	DSI_TCLKPOST: Time to drive clock after the last HS data
15:8	X	DSI_TCLKZERO: Time to drive LP00 before HS clock
7:0	X	DSI_TTLPX: LP period

26.10.3 DSI_DSI_PHY_TIMING_2_0

DSI D-PHY Timing Register 2

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
23:16	X	DSI_TCLKPREPARE: Time to drive LP0 before CLK_ZERO starts off on the clock lane. **Note:** Valid programmable values for the following register fields (i.e., DSI_TCLKPRE/ DSI_TCLKPREPARE) are 0 to 254. Invalid value is 255
15:8	X	DSI_TCLKPRE: Time to run clock before enabling data lane
7:0	X	DSI_TWAKEUP: LP period when exiting ULPM, in units of 512 byte clocks.

26.10.4 DSI_DSI_BTA_TIMING_0

DSI D-PHY Bus-Turn-Around Timing

Offset: 0x3f | Byte Offset: 0xfc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_TPKTBTA: Programmable Time delay between end of Host packet transmission and generation of Pkt BTA in PKT_BTA mode (i.e., HOST_DSI_CONTROL[2]) - Generate BTA at the end of Host packets **Note**: Valid programmable values for all the fields in this register (i.e., DSI_TTAGO/ DSI_TTASURE/DSI_TTAGET/ DSI_TPKTBTA) are 0 to 254. Invalid value is 255.
23:16	X	DSI_TTAGET: Time to Drive LP00 at end of BTA (5 * TTLPX)
15:8	X	DSI_TTASURE: Time to Receive LP00 at end of BTA (2 * TTLPX)
7:0	X	DSI_TTAGO: Time to drive LP00 at start of BTA (4 * TTLPX)

26.11 Contention Recovery Timers

These registers control the length of time - in units of 512 DSI byte clocks – that can elapse before the hardware will decide that a bus contention error has occurred.

There are several counters to deal with various forms of error that may occur. For details, refer to the MIPI DSI Specification, Section 7.2.2.

26.11.1 DSI_DSI_TIMEOUT_0_0

DSI Time Out Terminal Count Register 0

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LRXH_TO: Low Power Receive (Host) Time Out terminal count
15:0	X	HTX_TO: High Speed Transmit Time Out terminal count

26.11.2 DSI_DSI_TIMEOUT_1_0

DSI Time Out Terminal Count Register 1

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PR_TO: Peripheral Reset duration.
15:0	X	TA_TO: Turn Around Time Out terminal count

26.11.3 DSI_DSI_TO_TALLY_0

DSI Time Out Tally Register

Each time one of the time-out counters reaches its terminal count, it will increment the associated tally register.

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	R/W	Reset	Description
24	RO	X	P_RESET_STATUS: Peripheral Reset time out status 0 = IN_RESET 1 = READY
23:16	RW	X	TA_TALLY: Turn Around time out tally
15:8	RW	X	LRXH_TALLY: LP Rx time out tally
7:0	RW	X	HTX_TALLY: HS Tx time out tally

26.12 Physical Pad Control Registers

26.12.1 DSI_PAD_CONTROL_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x4b | Byte Offset: 0x12c | Read/Write: R/W | Reset: 0x010f010f (0bxxxxxxx1xxxx1111xxxxxxx1xxxx1111)

Bit	Reset	Description
24	0x1	DSI_PAD_PULLDN_CLK_ENAB:1= Enable pad pulldown for clock bit at power on
19:16	0xf	DSI_PAD_PULLDN_ENAB:1= Enable pad pulldown for data bits at power on
8	0x1	DSI_PAD_PDIO_CLK: Power down for clock bit, drivers, receivers, and contention detectors
3:0	0xf	DSI_PAD_PDIO: Power down for data bit, drivers, receivers, and contention detectors

26.12.2 DSI_PAD_CONTROL_CD_0

Contention Detection Logic Enable Signals

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx000xxxxxxx0xxxx0000)

Bit	Reset	Description
18:16	0x0	DSI_PAD_CDDNADJ: Level adjust on low limit of detection. Tie to 0 (DSI emu, miniTMC control). 000 = 0.3V 001 = 0.375V 010 = 0.45V 011 = 0.525V 100 = 0.3V 101 = 0.225V 110 = 0.15V 111 = 0.075V
8	0x0	DSI_PAD_CD_EN_CLK: Clock bit contention detector enable. 1 = Enable
3:0	0x0	DSI_PAD_CD_EN: Data bits contention detector enable. 1 = EnableCD_EN_[1] = 0 in DSI.

26.12.3 DSI_PAD_CD_STATUS_0

Contention Detection Status from MIPI PAD

Offset: 0x4d | Byte Offset: 0x134 | Read/Write: RO | Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	DSI_PAD_CDN_CLK
16	X	DSI_PAD_CDP_CLK
11:8	X	DSI_PAD_CDN
3:0	X	DSI_PAD_CDP

26.12.4 DSI_DSI_VID_MODE_CONTROL_0

Host Command Packet During Video Mode

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:1	0x0	DSI_LINE_TYPE: Vertical blank. LINE TYPE on which the host command packet is to be transmitted, i.e., valid values are 0/1/2/4 0 = ZERO: Line type 0 1 = ONE: Line type 1 2 = TWO: Line type 2 3 = THREE: NA: Host command packets on line type 3 are invalid 4 = FOUR: Line type 4 5 = FIVE: NA: Host command packets on line type 5 are invalid 6 = SIX: NA: Host command packets on line type 6 are invalid 7 = SEVEN: NA: Host command packets on line type 7 are invalid
0	0x0	DSI_CMD_PKT_VID_ENABLE: 0 = DISABLE: Disable host command packet during video mode 1 = ENABLE: Enable host command packet during video mode

26.12.5 DSI_PAD_CONTROL_1_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000x000x000x000)

Bit	Reset	Description
14:12	0x0	DSI_PAD_OUTADJ3: Input delay trimmer for data bit 3. Each tap delays 40 ps

Bit	Reset	Description
10:8	0x0	DSI_PAD_OUTADJ2: Input delay trimmer for data bit 2. Each tap delays 40 ps
6:4	0x0	DSI_PAD_OUTADJ1: Input delay trimmer for data bit 1. Each tap delays 40 ps
2:0	0x0	DSI_PAD_OUTADJ0: Input delay trimmer for data bit 0.Each tap delays 40 ps

26.12.6 DSI_PAD_CONTROL_2_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx000x000x000x000x000)

Bit	Reset	Description
18:16	0x0	DSI_PAD_SLEWUPADJ: Pull-up slew rate adjust. 000 = 011: slew rate increases 100 = 000, 100 =111: skew rate decreases
14:12	0x0	DSI_PAD_SLEWDNADJ: Pull-down slew rate adjust. 000 = 011, slew rate increases 100 = 000, 100 = 111: skew rate decreases.
10:8	0x0	DSI_PAD_LPUPADJ: Driver pull-up impedance control. 00 = 130 ohms, default, 01 = 110 ohms 10 = 130 ohms 11 = 150 ohms
6:4	0x0	DSI_PAD_LPDNADJ: Driver pull-down impedance control. 00 = 130 ohms, default 01 = 110 ohms 10 = 130 ohms, same as 0 011 = 150 ohms
2:0	0x0	DSI_PAD_OUTADJCLK: Output trimmer delay for clock bit. Each tap delays 40 ps

26.12.7 DSI_PAD_CONTROL_3_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: R/W | Reset: 0x10000000 (0bxxx1xxxxxxxxxxx0xx00xx00xx00xx00)

Bit	Reset	Description
28	0x1	DSI_PAD_PDVCLAMP: Power down regulator which supplies current to serializer/ deserializer logic. Active high, 1=power down
16	0x0	DSI_PAD_BANDWD_IN: Increase bandwidth of differential receiver
13:12	0x0	DSI_PAD_PREEMP_PD_CLK: Clock bit HS driver pull down pre-emphasis. 00 = no_preemphasis 11= max
9:8	0x0	DSI_PAD_PREEMP_PU_CLK: Clock bit HS driver pull up pre-emphasis. 00 = no_preemphasis 11= max
5:4	0x0	DSI_PAD_PREEMP_PD: Data bit HS driver pull down pre-emphasis. 00 = no_preemphasis 11= max
1:0	0x0	DSI_PAD_PREEMP_PU: Data bit HS driver pull up pre-emphasis. 00 = no_preemphasis 11= max

26.12.8 DSI_PAD_CONTROL_4_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxxx0000xxx0xxxx0000xxx0xxx0)

Bit	Reset	Description
28	0x0	DSI_PAD_HS_BSO_CLK:1= Enables BIAS and power regulators on for HS mode
23:20	0x0	DSI_PAD_HS_BSO:1= Enables BIAS and power regulators on for HS mode
16	0x0	DSI_PAD_LP_BSO_CLK:1= Enables BIAS and power regulators on for LP mode
11:8	0x0	DSI_PAD_LP_BSO:1= Enables BIAS and power regulators on for LP mode
4	0x0	DSI_PAD_TXBW_EN:1= Increase bandwidth of output driver, default=0
0	0x0	DSI_PAD_REV_CLK:1= Reverse clock polarity

26.12.9 DSI_DSI_GANGED_MODE_CONTROL_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1000)

Bit	Reset	Description
2:1	0x0	DUMMY_PIX_LEFT_RIGHT_SIDE: 0 = NORMAL: 0 - Dummy pixels not-enabled/Disabled on either left-Right side of active pixel stream 1 = LEFT : 1 - Dummy pixels enabled to the left side of active pixel stream 2 = RIGHT: 2 - Dummy pixels enabled to the Right side of active pixel stream 3 = NA: NA - Not supported
0	0x0	DSI_GANGED_MODE_EN: 0 = DISABLE: Ganged mode transaction disabled 1 = ENABLE: Ganged mode transaction enabled

26.12.10 DSI_DSI_GANGED_MODE_START_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	DSI_GANGED_START_POINTER: Start pointer for indicating the start of partial active valid pixel data to be latched from the valid pixels of the display controller in Left-Right/ Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non ganged mode).

26.12.11 DSI_DSI_GANGED_MODE_SIZE_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	DSI_GANGED_VALID_LOW_WIDTH: Width of Partial Inactive/Ignored pixel data from the valid pixels of the display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non ganged mode).
12:0	0x0	DSI_GANGED_VALID_HIGH_WIDTH: Width of Partial Active valid pixel data latched from the valid pixels of display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non ganged mode).

26.12.12 DSI_DSI_RAW_DATA_BYTE_COUNT_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DSI_RAW_DATA_BYTE_COUNT: Host RAW DATA byte count specifies the total number of bytes to send when "HOST_DSI_CONTROL[6] -> RAW_DATA" enabled

26.12.13 DSI_DSI_ULTRA_LOW_POWER_CONTROL_0

Ultra Low Power Sequence Control Register

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:8	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE3: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
7:6	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE2: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
5:4	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE1: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
3:2	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE0: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
1:0	0x0	DSI_ULTRA_LOW_POWER_CLK_LANE: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM

26.12.14 DSI_DSI_INIT_SEQ_DATA_8_0

DSI Init Sequence Write Data 8

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_8: DSI Init Sequence Write Data bits 31:0

26.12.15 DSI_DSI_INIT_SEQ_DATA_9_0

DSI Init Sequence Write Data 9

Offset: 0x59 | Byte Offset: 0x164 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_9: DSI Init Sequence Write Data bits 31:0

26.12.16 DSI_DSI_INIT_SEQ_DATA_10_0

DSI Init Sequence Write Data 10

Offset: 0x5a | Byte Offset: 0x168 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_10: DSI Init Sequence Write Data bits 31:0

26.12.17 DSI_DSI_INIT_SEQ_DATA_11_0

DSI Init Sequence Write Data 11

Offset: 0x5b | Byte Offset: 0x16c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_11: DSI Init Sequence Write Data bits 31:0

26.12.18 DSI_DSI_INIT_SEQ_DATA_12_0

DSI Init Sequence Write Data 12

Offset: 0x5c | Byte Offset: 0x170 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_12: DSI Init Sequence Write Data bits 31:0

26.12.19 DSI_DSI_INIT_SEQ_DATA_13_0

DSI Init Sequence Write Data 13

Offset: 0x5d | Byte Offset: 0x174 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_13: DSI Init Sequence Write Data bits 31:0

26.12.20 DSI_DSI_INIT_SEQ_DATA_14_0

DSI Init Sequence Write Data 14

Offset: 0x5e | Byte Offset: 0x178 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_14: DSI Init Sequence Write Data bits 31:0

26.12.21 DSI_DSI_INIT_SEQ_DATA_15_0

DSI Init Sequence Write Data 15

Offset: 0x5f | Byte Offset: 0x17c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_15: DSI Init Sequence Write Data bits 31:0

26.12.22 DSI_DUMMY_PIX_CNT_0

Offset: 0x60 | Byte Offset: 0x180 | Read/Write: R/W | Reset: 0x00XX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	RIGHT_DUMMY_PIX_CNT: Number of dummy pixels padded to the right of the active pixel stream i.e., [Min,Max] = [0,252] 0-> 1 pixels 251->252 pixels
7:0	X	LEFT_DUMMY_PIX_CNT: Number of dummy pixels padded to the left of the active pixel stream i.e., [Min,Max] = [0,252] 0-> 1 pixels 251->252 pixels

26.12.23 DSI_DSI_DSC_CONTROL_0

Note: *This feature is not supported.*

Display stream compression control register

Offset: 0x61 | Byte Offset: 0x184 | Read/Write: R/W | Reset: 0x00000300 (0bxxxxxxxxxxxxxxx00xxxx0011000000x0)

Bit	Reset	Description
17:16	0x0	NUM_COMPRESS_PKTS_PER_ROW: Number of compressed image packets per row between two sync events. Multiple packets option is for Video mode only. Always program ONE packet for Command mode. 0 = ONE: 1 compressed image packet for one slice per row 1 = TWO: 2 compressed image packets for two slices per row 2 = THREE: Reserved 3 = FOUR: 4 compressed image packets for four slices per row
11:2	0xc0	COMPRESS_RATE: Compression bit rate. Only 8/12/16 bpp compression rates are supported. Remaining values in this bit field are reserved. 128 = BPP_8 192 = BPP_12 256 = BPP_16
0	0x0	COMPRESS_MODE_EN: Enables compressed bitstream transport mode 0 = DISABLE: compression mode disabled 1 = ENABLE: compression mode enabled

CHAPTER 27: DISPLAY INTERFACES: HDMI AND DISPLAYPORT

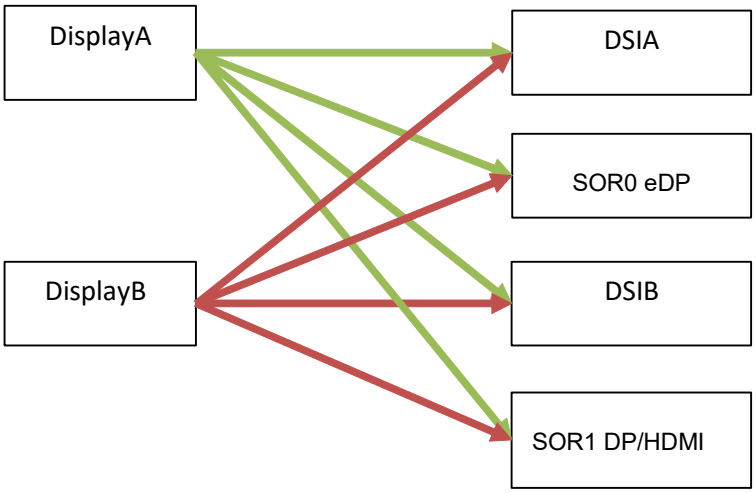
27.1 Overview

The Tegra X1 processor has two display serial output resources (SORs), which act as display interfaces: SOR0 and SOR1. They collect pixels from the output of a display pipeline, format/encode them to desired format, and then stream the data to various output devices. The SOR consists of several individual resources which can be used to interface with different display devices such as HDMI or DP. Each SOR can drive only a single device at any given time and can be connected to either display controller (Head).

SOR0 is for internal panels (no audio), and SOR1 is for either internal or external panels.

The Tegra X1 processor also has two MIPI-DSI interfaces, which support up to 2 x 4-lane mode. Refer to the Chapter 26: Display Interfaces: MIPI-DSI in this TRM for more information.

Figure 113: Output Connections



27.2 Features

SOR0 supports internal panels only, using eDP. Its features include:

- 1/2/4 lane eDP (eDP 1.4)
- Supports frame packed 3D stereo mode (not frame-sequential mode like dGPU)
- Supports generic infoframe
- ASSR copy protection
- 24-bit RGB pixel format only
- Supports up to HBR2 per DP 1.2 specification
- Supports eDP 1.4 features:
  - additional link rates (2.16, 2.43, 3.24, 4.32 Gbps)
  - enhanced framing
  - power sequencing
  - reduced auxiliary timing

- reduced main voltage swing

SOR1 supports DP and HDMI as well as eDP. Its features include:

- 1/2/4 lane DP (DP1.2) and eDP (eDP 1.4)
- HDMI 2.0, up to 594 MHz pixel clock
- HDCP 2.2 and 1.3 over either HDMI or DP
- ASSR (alternate seed scrambler reset) for internal eDP panels
- On HDMI, multichannel audio from HDA controller, up to 8 channels, 192 kHz, 24-bit.
- Supports frame-packed 3D stereo mode (not frame-sequential mode like dGPU)
- Supports generic infoframe transmission
- Supports HDMI Vendor Specific Infoframe (VSI) packet transmission
- Supports up to HBR2 link rate, per DP 1.2 specification
- Supports fuse calibration information for HDMI analog parameter(s)
- 24-bit RGB and 24-bit YUV444 (HDMI) pixel formats (YUV444 is mostly for 1080i support)
- 1080i output on HDMI (mostly implemented in the display controller)
- Supports DP or HDMI connectors via appropriate external level shifting
- Supports external Dual Mode standard (DP2HDMI passive or active adapters and adapter discovery)
- Supports eDP 1.4 features:
  - additional link rates (2.16, 2.43, 3.24, 4.32 Gbps)
  - enhanced framing
  - power sequencing
  - reduced auxiliary timing
  - reduced main voltage swing

27.3 eDP

The Tegra® X1 eDP block is a display output controller, supporting Embedded DisplayPort (eDP). It drives local panels only and does not support an external DP port. The eDP block has a pixel bus as an input from the display pipe of each head. It also has a small test pattern generator and formatter to encode pixels for specific output devices, a CRC generator, and a bundle interface to the display software interface.

The eDP block collects pixels from the output of the display pipeline, formats/encodes them to the eDP format, and then streams them to various output devices. The eDP block consists of individual resources that can interface with different display devices.

27.3.1 Features

These features are supported in the eDP block:

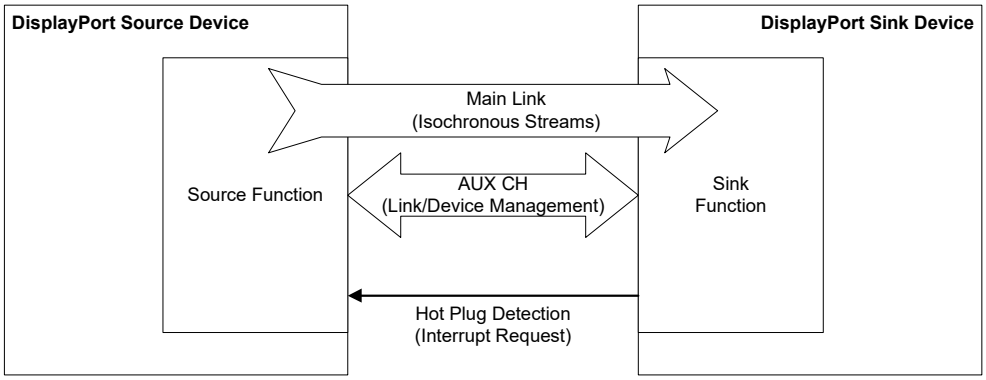
- eDP 1.4
  - 1/2/4 lanes, single link
  - RBR/HBR/HBR2
  - 18/24 bits color depth
  - Up to 540 MHz
  - Internal panel:

    - 4096 x 2160 @ 60Hz (2D – portrait/landscape)
    - 1920 x 1080 @ 60Hz (3D – portrait/landscape)
  - -0.5% down spread support
- Stereo mode
- Generic infoframe
- Supports DP AUX and HPD
- Single PHY to output eDP
  - 4-lane eDP
  - Supports 5.4 GHz eDP

27.3.2 DisplayPort Overview

The DisplayPort link consists of Main Link, Auxiliary Channel (AUX CH), and Hot Plug Detect (HPD) channels. Main Link is a unidirectional, high-bandwidth, and low-latency channel used for transport of isochronous streams such as uncompressed video and audio. Auxiliary Channel is a half-duplex, bidirectional channel used for link management and device control. The HPD signal serves as an interrupt request by the sink device. Main Link consists of AC-coupled, doubly terminated differential pairs (called lanes). Three link rates are supported: 5.4 Gbps, 2.7 Gbps, and 1.62 Gbps per lane. The Main Link supports 1, 2, or 4 lanes.

Figure 114: DisplayPort Link



27.4 HDMI

- HARDWARE_N feature enabled
- TMDS I/O macro with controls
- Interlaced video format support
- YUV444 output (BT.601 and BT.709 formats)
- HDMI in power-gated partition SOR
- Output range scaling for RGB
- Vendor-specific infoframe supported
- Audio features for HD audio include:
  - 2, 6, or 8 channels of LPCM (i.e., uncompressed linear PCM)
  - 16, 20, or 24 bits per sample
  - 32, 44.1, 48, 88.2, 96, 176.4, 192 K samples/second

27.5 HDCP

HDCP requires the presence of a correctly programmed KFUSE. Two main sequences are downstream authentication and upstream authentication. Downstream authentication permits encryption of the HDMI output, and involves a handshake between the HDMI module and the display device. Upstream authentication is more of a software convention between the application and HDMI driver, although the HDMI module does perform KFUSE operations to assist.

Communication with the downstream receiver uses DDC, which requires I^2C.

27.5.1 HDCP KFUSE Key Copying

HDCP keys are stored encrypted in the KFUSE module. Before starting HDCP, software must copy the keys (576 bytes) from the KFUSE to the HDMI registers.

Note: *KFUSE is just for HDCP 1.x. HDCP 2.0 does not use KFUSE.*

1. To enable use of KFUSE keys, set SOR_NV_PDISP_KEY_CTRL.LOCAL_KEYS = ENABLED.
2. Bring the KFUSE module out of reset and enable the clock, if not already enabled. This can be before or after HDMI is out of reset.
3. Wait for the KFUSE to read the keys from the fuse to its SRAM and perform error correction and CRC checking:
   a. Poll for KFUSE.STATE.DONE == 1
   b. Check KFUSE.STATE.CRCPASS == 1
   c. Check that SOR_NV_PDISP_KEY_CTRL.PKEY_LOADED == TRUE
4. Prepare to copy:
   a. KFUSE.KEYADDR.ADDR = 0
   b. KFUSE.KEYADDR.AUTOINC = 1
5. For each line in 0 .. 35 # 576 bytes = 144 words / 4 words-per-line = 36 lines
   a. For each word in 0..3:

      data[word] = KFUSE.KEYS # auto-increments read address
   b. SOR_NV_PDISP_KEY_HDCP_KEY_0 = data[0]
   c. SOR_NV_PDISP_KEY_HDCP_KEY_1 = data[1]
   d. SOR_NV_PDISP_KEY_HDCP_KEY_2 = data[2]
   e. SOR_NV_PDISP_KEY_HDCP_KEY_3 = data[3]
   f. SOR_NV_PDISP_KEY_HDCP_KEY_TRIG.LOAD_HDCP_KEY = TRIGGER # trigger decryption of 16 bytes
   g. If line == 0: SOR_NV_PDISP_KEY_CTRL.AUTOINC = DISABLED. Else:
      SOR_NV_PDISP_KEY_CTRL.AUTOINC = ENABLED
   h. SOR_NV_PDISP_KEY_CTRL.WRITE16 = TRIGGER # wait for decrypt, and copy decrypted 16 bytes to the local key store.
   i. Poll for SOR_NV_PDISP_KEY_CTRL.WRITE16 == DONE
6. Write SOR_NV_PDISP_KEY_SKEY_INDEX.IDX_VALUE with the AES key selector that was used to encrypt the keyglob data. This is currently fixed at 0.

27.5.2 Downstream Authentication

Note: *For register names containing "DP/TMDS", use DP in DP-HDCP mode and TMDS in HDMI-HDCP mode.*

This can be done before enabling transmission.

1. Set SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.ONEONE = ENABLE. SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.RUN = YES. Causes the RTL to access HDCP keys and compute AN and other parameters.
2. Poll for SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.AN = VALID
3. Read Aksv from SOR_NV_PDISP_SOR_DP/TMDS_HDCP_AKSV_LSB/MSB and check validity (exactly 20 of 40 bits = 1). If invalid, the KFUSE might be blank.
4. Read Bksv from the receiver (via I2C), check validity (20 high bits), and write it into SOR_NV_PDISP_SOR_DP/TMDS_HDCP_BKSV_LSB/MSB.
5. Read AN from SOR_NV_PDISP_SOR_DP/TMDS_HDCP_AN_LSB/MSB, and write to the receiver via I^2C.
6. Read Aksv from SOR_NV_PDISP_SOR_DP/TMDS_HDCP_AKSV_LSB/MSB, check validity (20 high bits), and write to receiver via I^2C.
7. Poll for SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.R0 = VALID
8. Read SOR_NV_PDISP_SOR_DP/TMDS_HDCP_RI and compare with the receiver's RI. They should match.

If the above procedure is successful, then SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.CRYPT can be set to ENABLE.

To verify that encryption is enabled, check that SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.RUN == YES.

27.5.3 Upstream Authentication

The HDMI module does not do the protocol itself. Rather, it provides primitives that software can use. These primitives access HDCP keys and status, and compute values.

Typically, downstream authentication has been completed before doing the upstream.

The "Upstream Setup" sequence is:

1. Set SOR_NV_PDISP_SOR_TMDS_HDCP_CMODE.MODE = READ_S
2. Poll for SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.SPRIME = INVALID
3. Program SOR_NV_PDISP_SOR_TMDS_HDCP_CN_LSB/MSB and SOR_NV_PDISP_SOR_TMDS_HDCP_CKSV_LSB/MSB with Cn and Cksv. NOTE: CKSV_MSB must be last since it is a trigger.
4. Poll for SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.SPRIME = VALID
5. Read Dksv from SOR_NV_PDISP_SOR_TMDS_HDCP_DKSV_LSB/MSB and verify validity
6. Read Cs from SOR_NV_PDISP_SOR_TMDS_HDCP_CS_LSB/MSB
7. Read S' from SOR_NV_PDISP_SOR_TMDS_HDCP_SPRIME_LSB1/LSB2/MSB

Software uses these read values to perform the authentication handshake with application software.

The "Upstream M" sequence is:

1. Set SOR_NV_PDISP_SOR_TMDS_HDCP_CMODE.MODE = READ_M.
2. Poll for SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.MPRIME = INVALID.
3. Program SOR_NV_PDISP_SOR_TMDS_HDCP_CN_LSB/MSB and SOR_NV_PDISP_SOR_TMDS_HDCP_CKSV_LSB/MSB with Cn and Cksv. NOTE: CKSV_MSB must be last since it is a trigger.
4. Poll for SOR_NV_PDISP_SOR_DP/TMDS_HDCP_CTRL.MPRIME = VALID.
5. Read Dksv from SOR_NV_PDISP_SOR_TMDS_HDCP_DKSV_LSB/MSB and verify validity.
6. Read M' from SOR_NV_PDISP_SOR_TMDS_HDCP_MPRIME_LSB/MSB.

27.6 SOR Clocking

PLLDP used for eDP and DP link clock supports spread-spectrum.

PLLD2, used for the HDMI link and pixel clocks and the DP pixel clock, supports fractional-N for fine frequency control. The Fractional-N can be changed glitchlessly on the fly.

MAUD_CLK and HDA2CODEC_2X_CLK are used for audio and must remain synchronous; they can be driven from either PLLP or PLLA. PLLA allows fine frequency control (Fractional-N) to allow software to match the audio sample rate to an external broadcast stream.

I/O Cell/Pad clock input is selectable and depends on the protocol that is selected for output. For TMDS, this clock is equal to pclk, and for DP it is equal to 270 MHz PLLDP.

For both DP and TMDS, it is required to use the feedback clock from the macro for the digital logic. A glitchless 2:1 MUX needs to be added; one input is the feedback clock from the macro and is a fixed “safe” clock that is guaranteed to be running.

The feedback clock is needed because:

- For eDP/DP, the macro generates the clock that needs to be used by the SOR
- For TMDS, the PLL output and the macro clock might not be aligned in phase and requires “LOADADJ” calibration
- For some modes, the macro reference clock should be different from the pixel clock, for better performance

The following software programming sequence is recommended for clocks:

- Program the MUX to a safe clock
- Program/enable the PLLD2 or PLLDP reference PLL
- Program various macro registers to bring up the macro clock based on eDP/DP or TMDS mode. All of the macro registers are in the SOR.
- Program the MUX select to SOR clock.
- Program the rest of display to connect to the SOR.

27.6.1 DP Mode

The frequency of the SOR clock depends on the protocol, link training, pixel depth, and pixel replication. For DP, the SOR clock is generated in the SOR analog macro. The analog macro uses a 270 MHz differential reference clock from PLLDP. The macro multiplies this clock by 6, 10, or 20 to produce 1.62 GHz, 2.7 GHz, or 5.4 GHz link clock, respectively. The link clock is divided by 10 to produce the low speed SOR clock of 162 MHz, 270 MHz, or 540 MHz. The link clock frequency is chosen based on the bandwidth requirements of the stream, as described below.

The DP link can operate in 1, 2, or 4 lane mode and from 1.62 GHz to 5.4 GHz. In order to calculate the available bandwidth on the DP link, use the following equation:

```
Available Bandwidth = Link Rate/10 * bits per clock * number of lanes * 0.995
```

Link Rate/10 is the speed at which the SOR runs while in DP mode (for corner cases, you may need to account for spread). bits_per_clock is always 8. The number of lanes is 1, 2, or 4. In order to calculate the amount of bandwidth needed for a resolution, use the following equation:

```
Required Bandwidth = pixel clock * bits per pixel
```

If the Required Bandwidth is less than the Available Bandwidth, then that resolution can be supported at that link configuration. For example:

- 3840x2400x24bpp @60 Hz, 592.5 MHz pclk
  4 Lanes, 5.4 GHz

```
Available Bandwidth = 5400 MHz / 10 * 8 * 4 * 0.995 = 17193.6 Mbps
Required Bandwidth = 592.5 * 24 = 14220 Mbps
```

This resolution is supported at this link rate.

- 4096x2560x24bpp @60 Hz, 672.0 MHz pclk
  4 Lanes, 5.4 GHz

```
Available Bandwidth = 5400 MHz / 10 * 8 * 4 * 0.995 = 17193.6 Mbps
Required Bandwidth = 672 * 24 = 16128 Mbps
```

This resolution is supported at this link rate.

27.6.2 TMDS Mode

For single link TMDS (DVI/HDMI), the SOR clock and link clock are equal to the pixel clock frequency, because Tegra X1 does not support deep color or pixel replication. The I/O Cell generates the SOR clock by multiplying its pixel clock reference by 10 or 20, then dividing by 10.

27.6.3 Link Training

Link training can cause the link speed or lane count to change. Whenever OR clocks are changed, the Display Controller needs to be disconnected from the OR so that the OR pipeline can go idle. Changing the clocks while the OR pipeline is actively encoding pixels can leave the pipe in an unknown state. There are also many DP parameters that are dependent on the link rate, so if the link rate is changed on the fly, it could cause FIFO overflows/underflows and possibly hangs.

27.6.4 M_VID

In DP mode, the SOR1 must derive the ratio of link clock and pixel clock, so that the DP receiver can recover the pixel clock.

Each display head sends a version of its pixel clock to the SOR, used to transfer pixel data: display2sor1_clk, displayb2sor1_clk.

The M_VID ratio is derived separately for each display head. Then the correct head value is muxed along with the head's pixel data.

27.6.5 Audio Timestamp

In DP mode, the SOR1 must derive the ratio of link clock to audio sample clock. In Tegra X1 devices, the SOR logic uses a fixed 102 MHz clock called MAUD_CLK for this; derived from the same PLL as HDA2CODEC_2X clock uses. The 102 MHz MAUD_CLK must track the 48 MHz HDA2CODEC_2X_CLK (and thus the actual sample rate) and maintain the exact 2.125 ratio. That implies using the same PLL source (PLLP or possibly PLLA) for both.

Software programs N and D values to generate a 512*fs clock from the fixed 102 MHz clock:

512*fs / 102 = N/D

For 48 KHz, 512*48000/102000000 = 512/2125.

Thus, every 102 * 512/2125 MHz, the 512*fs counter increments. Every Naud=2^15 link clocks, the counter value is latched, giving Maud.

The following table shows the nominal N/D values for supported audio frequencies:

Table 162: MAUD N/D

Maudclk Freq	48 kHz		96 kHz		192 kHz		44.1 kHz		88.2 kHz		176.4 kHz	
	N	D	N	D	N	D	N	D	N	D	N	D
102 MHz	512	2125	1024	2125	2048	2125	2352	10625	4704	10625	9408	10625

The audio sample rate (and thus MAUD_CLK) need not be a fixed synchronous ratio to the link clock; this would be the case in spread-spectrum or frequency-hopping link clock, or when audio PLL uses fractional-N to lock to external source. In such cases, the Maud will vary from the nominal value. For example, if link clock slows down 0.5%, the computed Maud will increase by 0.5 %. If audio PLL slows down by 0.1% for 48 * 1000/1001 KHz sampling, the computed Maud will decrease by 0.1%.

27.6.6 SOR Clock Switch

The SOR clock should be disabled while doing SOR clock switching.

For SOR0:

- CLK_RST_CONTROLLER.CLK_OUT_ENB_X.CLK_ENB_SOR0 = DISABLE
- CLK_RST_CONTROLLER.CLK_SOURCE_SOR0.SOR0_CLK_SEL1 = …
- CLK_RST_CONTROLLER,CLK_SOURCE_SOR0.SOR0_CLK_SEL0 = …
- CLK_RST_CONTROLLER.CLK_OUT_ENB_X.CLK_ENB_SOR0 = ENABLE

SOR1 has similar settings.

27.7 Programming Guidelines

This subsection provides SOR programming guidance for software. It covers the following topics:

- The programming sequence for SOR start: attaching and powering up SOR (eDP/DP and HDMI)
- The programming sequence for SOR stop: detaching and powering down SOR (eDP/DP and HDMI)
- The programming requirements for display standard raster timing used in SOR

27.7.1 DP Mode

For eDP/DP without audio, programming is the same between SOR0 and SOR1. All the programming below refers to SOR. The same is applicable to SOR1 and the DPAUX modules as well. Controls in CAR and Display Controllers are mentioned explicitly.

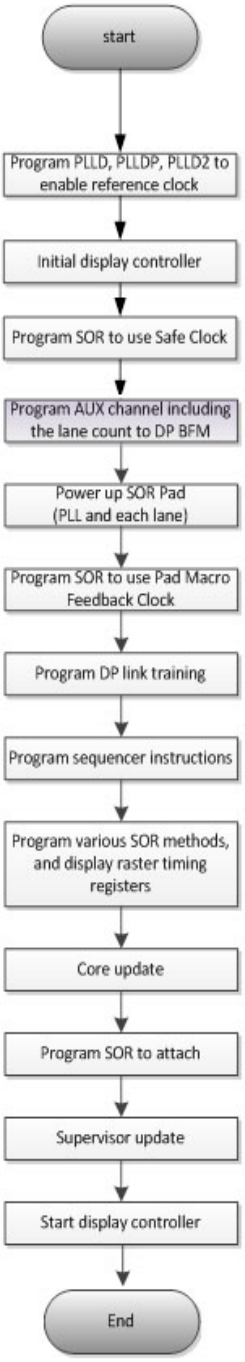
Controls in Display

```
DISP_WIN_OPTIONS.SOR_ENABLE ENABLE
DISP_WIN_OPTIONS.SOR1_ENABLE ENABLE
```

Controls in CAR

```
CLK_OUT_ENB_X.CLK_ENB_SOR0
RST_DEV_X_SET.SET_SOR0_RST
CLK_OUT_ENB_X.CLK_ENB_SOR1
RST_DEV_X_SET.SET_SOR1_RST

CLK_OUT_ENB_X.CLK_ENB_DPAUX
RST_DEV_X_SET.SET_DPAUX_RST
CLK_OUT_ENB_Y.CLK_ENB_DPAUX1
RST_DEV_Y_SET.SET_DPAUX1_RST
```

27.7.1.1 SOR Start



27.7.1.2 eDP Start Sequence

Program PLLD□PLLDP and PLLD2 to Enable Reference Clock

Software has to program PLLD, PLLDP, and PLLD2 first to enable reference clock, because the reference clock should be stable before enabling the display controller and SOR.

Initialize the Display Controller

The Display Controller should be initialized before programming the SOR.

```
STATE_ACCESS.WRITE_MUX ACTIVE
```

Set display to use the active copy of shadow register. Any programming to display register will be active immediately.

```
DISP_WIN_OPTIONS.SOR_ENABLE ENABLE
DISP_WIN_OPTIONS.SOR1_ENABLE ENABLE
```

The above register controls the sor_enable signal in display which controls the enabling of display2sor clock gating and raster timing signals.

Program display pixel clock so dcpclk is valid. Program display window selection registers and window initial properties (can be overridden later).

The display pixel clock should be programmed to be valid at this stage. The pixel clock feeds to SOR pad CLOCKIN, which generates the CLOCKOUT that will be used later by the SOR as orclk in HDMI mode. Before the SOR switches to use pad CLOCKOUT as orclk, the CLOCKOUT should be stable.

```
WIN_OPTIONS.WIN_ENABLE ENABLE
Enables display window.
```

Program AUX Channel

Program AUX registers to control DP parameters.

Notes:

- *In Tegra X1, DPAUX pad macro's power mode is controlled by DPAUX register DPAUX_HYBRID_SPARE_0.PAD_PWR. The reset value of this register is POWER UP. Therefore the default power mode of DPAUX pad macro is POWER UP.*
- ***[For SW]** The programming on DPAUX_HYBRID_SPARE_0 register is not applicable to software. Software need use another approach to program the Aux channel.*

Power up SOR Pad

Power up PLLDP to generate the macro clock and start to run the lane sequencer.

The programming sequence should follow the SOR pad specification.

1. Wait at least 5 µs
2. De-assert PDBG

   ```
   SOR.NV_PDISP_SOR_PLL2.AUX6          BANDGAP_POWERDOWN_DISABLE
   ```

3. Wait 20 µs
4. De-assert PDPLL, PLLVCOPD, PLLCAPPD to enable PLL

   ```
   SOR.NV_PDISP_SOR_PLL3.PLLVDD_MODE    V3_3
   SOR.NV_PDISP_SOR_PLL0.PWR            ON
   SOR.NV_PDISP_SOR_PLL0.VCOPD          RESCIND
   SOR.NV_PDISP_SOR_PLL2.AUX1           SEQ_PLLCAPPD_OVERRIDE
   SOR.NV_PDISP_SOR_PLL2.AUX8           SEQ_PLLCAPPD_ENFORCE_DISABLE
   ```

5. Wait 200 µs till PLL is locked

   ```
   SOR.NV_PDISP_SOR_PLL2.AUX2           OVERRIDE_POWERDOWN
   SOR.NV_PDISP_SOR_PLL2.AUX7           PORT_POWERDOWN_DISABLE
   ```

6. Power up SOR macro lanes
7. Power up 1, 2 or 4 lanes

   1 lane:

   ```
   SOR.NV_PDISP_SOR_DP_LINKCTL0.LANECOUNT ONE
   ```

```
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_0 NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_1 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_2 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_3 YES
```

2 lanes:

```
SOR.NV_PDISP_SOR_DP_LINKCTL0.LANECOUNT TWO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_0 NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_1 NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_2 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_3 YES
```

4 lanes:

```
SOR.NV_PDISP_SOR_DP_LINKCTL0.LANECOUNT FOUR
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_0 NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_1 NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_2 NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_3 NO
```

8. Start lane sequencer

```
SOR.NV_PDISP_SOR_LANE_SEQ_CTL -dirty
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.DELAY             15
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.NEW_POWER_STATE   PU
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SEQUENCE          DOWN
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SETTING_NEW       TRIGGER
```

9. Select to use single clock for macro

```
SOR.NV_PDISP_SOR_CLK_CNTRL.DP_CLK_SEL SINGLE_DPCLK
```

10. Other programming

```
SOR.NV_PDISP_SOR_DP_LINKCTL0.ENABLE YES
SOR.NV_PDISP_SOR_DP_LINKCTL0.TUSIZE <specified by test arguments>
SOR.NV_PDISP_SOR_DP_LINKCTL0.ENHANCEDFRAME <specified by test arguments>
SOR.NV_PDISP_SOR_DP_SPARE0.SEQ_ENABLE YES
```

Switch to Use Pad Macro Feedback Clock

The SOR clock needs to be switched to connect to feedback clock from the macro. This switching is done by programming the registers in the CAR. Assuming that we want to use a PLLD source for PCLK and want to use the feedback clock as SORCLK, the following program can be used.

```
CLK_RST_CONTROLLER_CLK_SOURCE_SOR0.SOR0_CLK_SEL1 0
CLK_RST_CONTROLLER_CLK_SOURCE_SOR0.SOR0_CLK_SEL0 1
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SRC PLLD
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL1 0
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL0 1
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_DIVISOR
```

For SOR0, SOR0_CLK_SEL1 is optional. Since it supports only eDP, the only clocks are SAFECLK and FEEDBACK.

For SOR1, SOR1_CLK_SEL1 is used to mux between SAFECLK and HDMICLK in HDMI mode. The output of this mux is then muxed with FEEDBACK clock.

CLK_DIVISOR is used only in TMDS mode and is not available in SOR0.

Program Sequencer Instructions

Program SOR sequencer instructions to run and stop sequencer.

```
SOR.NV_PDISP_SOR_SEQ_CTL.PU_PC 0
SOR.NV_PDISP_SOR_SEQ_CTL.PU_PC_ALT 0
SOR.NV_PDISP_SOR_SEQ_CTL.PD_PC 8
SOR.NV_PDISP_SOR_SEQ_CTL.PD_PC_ALT 8
SOR.NV_PDISP_SOR_SEQ_CTL.SWITCH 0
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.DELAY 0
```

For non-sequencer tests, only program basic sequence without switching.

1. Sequencer up instructions

```
SOR.NV_PDISP_SOR_SEQ_INST0.WAIT_TIME 0
SOR.NV_PDISP_SOR_SEQ_INST0.WAIT_UNITS VSYNC
SOR.NV_PDISP_SOR_SEQ_INST0.HALT TRUE
SOR.NV_PDISP_SOR_SEQ_INST0.PIN_A LOW
SOR.NV_PDISP_SOR_SEQ_INST0.PIN_B LOW
SOR.NV_PDISP_SOR_SEQ_INST0.DRIVE_PWM_OUT_LO TRUE
SOR.NV_PDISP_SOR_SEQ_INST0.TRISTATE_IOS ENABLE_PINS
SOR.NV_PDISP_SOR_SEQ_INST0.BLACK_DATA NORMAL
SOR.NV_PDISP_SOR_SEQ_INST0.BLANK_DE NORMAL
SOR.NV_PDISP_SOR_SEQ_INST0.BLANK_H NORMAL
SOR.NV_PDISP_SOR_SEQ_INST0.BLANK_V NORMAL
SOR.NV_PDISP_SOR_SEQ_INST0.ASSERT_PLL_RESET NORMAL
SOR.NV_PDISP_SOR_SEQ_INST0.POWERDOWN_MACRO NORMAL
SOR.NV_PDISP_SOR_SEQ_INST0.LANE_SEQ RUN
SOR.NV_PDISP_SOR_SEQ_INST0.SEQUENCE UP
```

2. Sequencer down instructions

```
SOR.NV_PDISP_SOR_SEQ_INST8.WAIT_TIME 0
SOR.NV_PDISP_SOR_SEQ_INST8.WAIT_UNITS VSYNC
SOR.NV_PDISP_SOR_SEQ_INST8.HALT TRUE
SOR.NV_PDISP_SOR_SEQ_INST8.PIN_A LOW
SOR.NV_PDISP_SOR_SEQ_INST8.PIN_B LOW
SOR.NV_PDISP_SOR_SEQ_INST8.DRIVE_PWM_OUT_LO TRUE
SOR.NV_PDISP_SOR_SEQ_INST8.TRISTATE_IOS ENABLE_PINS
SOR.NV_PDISP_SOR_SEQ_INST8.BLACK_DATA NORMAL
SOR.NV_PDISP_SOR_SEQ_INST8.BLANK_DE NORMAL
SOR.NV_PDISP_SOR_SEQ_INST8.BLANK_H NORMAL
SOR.NV_PDISP_SOR_SEQ_INST8.BLANK_V NORMAL
SOR.NV_PDISP_SOR_SEQ_INST8.ASSERT_PLL_RESET NORMAL
SOR.NV_PDISP_SOR_SEQ_INST8.POWERDOWN_MACRO NORMAL
SOR.NV_PDISP_SOR_SEQ_INST8.LANE_SEQ RUN
SOR.NV_PDISP_SOR_SEQ_INST8.SEQUENCE DOWN
```

Before Attaching Program SOR Methods

Program the following SOR state control registers.

```
SOR.NV_PDISP_SOR_STATE1. ASY_OWNER
SOR.NV_PDISP_SOR_STATE1.ASY_SUBOWNER
SOR.NV_PDISP_SOR_STATE1.ASY_PROTOCOL
```

```
SOR.NV_PDISP_SOR_STATE1.ASY_HSYNCPOL
SOR.NV_PDISP_SOR_STATE1.ASY_VSYNCPOL
SOR.NV_PDISP_SOR_STATE1.ASY_DEPOL
SOR.NV_PDISP_SOR_STATE1.ASY_CRCMODE
SOR.NV_PDISP_SOR_STATE1.ASY_PIXELDEPTH
```

Program Display Raster Timing Registers

Program head control registers for hsync, vsync, hblank, vblank, htotal, vtotal.

Program these registers for each head connected.

```
SOR.NV_PDISP_HEAD_STATE0[NUM_HEADS].INTERLACED PROGRESSIVE
SOR.NV_PDISP_HEAD_STATE0[NUM_HEADS].DYNRANGE <specified by test arguments>
SOR.NV_PDISP_HEAD_STATE0[NUM_HEADS].COLORSPACE RGB
SOR.NV_PDISP_HEAD_STATE1[NUM_HEADS].VTOTAL <specified by test arguments>
SOR.NV_PDISP_HEAD_STATE1[NUM_HEADS].HTOTAL <specified by test arguments>
```

The Tegra X1 display has constraints on raster timing register configurations, therefore the programming to the SOR.NV_PDISP_HEAD_STATE registers should also be constrained to match with display.

```
if (REF_TO_SYNC.H_REF_TO_SYNC + REF_TO_SYNC.H_REF_TO_SYNC) < 20 {
SOR.NV_PDISP_HEAD_STATE1.HTOTAL $head [expr 19 - [REF_TO_SYNC.H_REF_TO_SYNC] + $total_width -
$blank_endx]
   SOR.NV_PDISP_HEAD_STATE3.HBLANK_END $head [expr 19 - [REF_TO_SYNC.H_REF_TO_SYNC]]
   SOR.NV_PDISP_HEAD_STATE4.HBLANK_START $head [expr 19 - [REF_TO
}


SOR.NV_PDISP_SOR_CSTM.ROTCLK 2
```

Attach and Update Methods

- Core update (sor_update)

```
SOR.NV_PDISP_SOR_STATE0.UPDATE 0
SOR.NV_PDISP_SOR_STATE0.UPDATE 1
SOR.NV_PDISP_SOR_STATE0.UPDATE 0
```

- Attach SOR

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
```

- Set OR mode to NORMAL

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      NORMAL
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
```

- Set Head OP mode to AWAKE

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE AWAKE
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      NORMAL
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
```

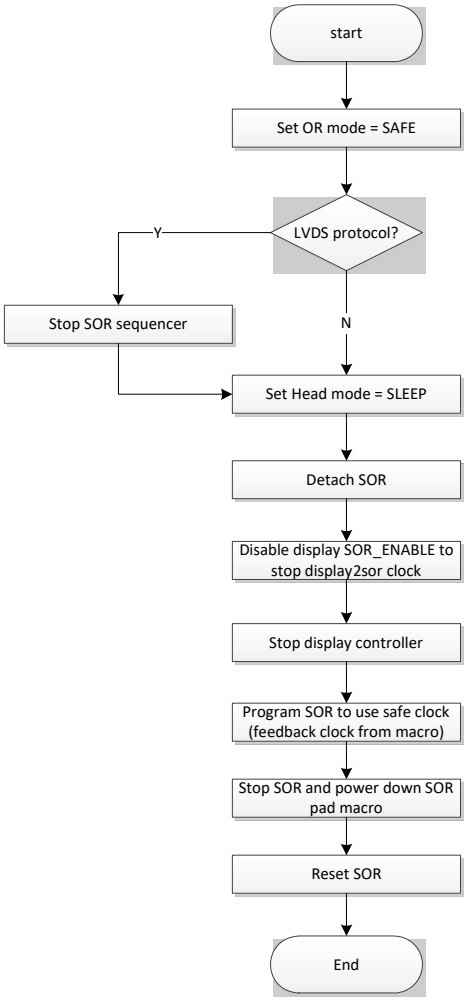
- Super update

```
SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 1
SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
```

Enable Display Controller

Now start the display controller by programming the display registers.

27.7.1.3 SOR Stop



27.7.1.4 eDP Stop Sequence

Set OR mode to SAFE

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE AWAKE
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
sor_super_update
proc sor_super_update {} {
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
    flush
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 1
    flush
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
    flush
}
```

Here OR mode can be polled for SAFE (SOR.NV_PDISP_SOR_PWR.MODE.SAFE).

Set Head Mode to SLEEP

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
sor_super_update
```

Detach SOR

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        NO
sor_super_update
Here can poll SOR TEST to detach (SOR.NV_PDISP_SOR_TEST.ATTACHED.FALSE)
SOR.NV_PDISP_SOR_STATE1.ASY_OWNER NONE
SOR.NV_PDISP_SOR_STATE1.ASY_SUBOWNER NONE
SOR.NV_PDISP_SOR_STATE1.ASY_PROTOCOL LVDS_CUSTOM
sor_update
proc sor_update {} {
    SOR.NV_PDISP_SOR_STATE0.UPDATE 0
    flush
    SOR.NV_PDISP_SOR_STATE0.UPDATE 1
    flush
    SOR.NV_PDISP_SOR_STATE0.UPDATE 0
    flush
}
```

Disable Display SOR_ENABLE to Stop display2sor Clock

Set display to use the active copy of shadow register; this enables the disabling of PW* registers to take effect immediately (otherwise it will wait until the frame end since display uses the assembly copy of registers).

```
STATE_ACCESS.WRITE_MUX ACTIVE
DISP_WIN_OPTIONS.SOR_ENABLE DISABLE //this will stop display2sor pixel clock
```

Stop the Display Controller

```
DISPLAY_COMMAND.DISPLAY_CTRL_MODE STOP // stop sending frame at the next frame boundary
DISPLAY_POWER_CONTROL.PW0_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW1_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW2_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW3_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW4_ENABLE DISABLE
```

Program SOR to Use Feedback Clock from Macro (Safe Clock)

Requires CAR to be programmed for the SOR to use feedback clock from the macro.

System test implements the register programming by function:

```
sor_clk_sel safe
```

Stop SOR and Power Down SOR Pad Macro

- Power down each lane of SOR
- Set PDPLL, PLLVCOPD, PLLCAPPD to 1
- Wait for at least 5 µs

- Power down PDBG: set PDBG to 1
- Wait for at least 5 µs
- Power down E_DPD: set E_DPD to 1

1. Power down each lane for eDP

```
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_0 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_1 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_2 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_3 YES
```

2. Stop lane sequencer

```
SOR.NV_PDISP_SOR_LANE_SEQ_CTL -dirty //enables config with the same value as is
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.DELAY             15
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.NEW_POWER_STATE   PD
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SEQUENCE          DOWN
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SETTING_NEW       TRIGGER
```

3. Here can poll SOR lane sequencer control status for SETTING_NEW done (SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SETTING_NEW.DONE)
4. Power down PDPORT to power down all the output links and lanes

```
SOR.NV_PDISP_SOR_PLL2.AUX7          PORT_POWERDOWN_ENABLE
```

5. Wait for 20 µs
6. Power down PDPLL, PLLVCOPD, PLLCAPPD

```
SOR.NV_PDISP_SOR_PLL0.PWR           OFF
SOR.NV_PDISP_SOR_PLL0.VCOPD         ASSERT
SOR.NV_PDISP_SOR_PLL2.AUX1          SEQ_PLLCAPPD_OVERRIDE
SOR.NV_PDISP_SOR_PLL2.AUX8          SEQ_PLLCAPPD_ENFORCE_ENABLE
```

7. Wait for 20 us
8. Power down PDBG

```
SOR.NV_PDISP_SOR_PLL2.AUX6          BANDGAP_POWERDOWN_ENABLE
```

9. Set E_DPD to 1 (PWR)

Reset SOR

Tegra X1 requires SOR to be reset SOR after power down because after SOR is stopped, display's hvtgen does not clean the counters. Therefore if SOR is restarted, the counters will continue to count and result in a function error.

The reset is implemented by programming the following CAR register (Reset device SOR0).

```
CLK_RST_CONTROLLER_RST_DEVICES_X
```

27.7.2 HDMI

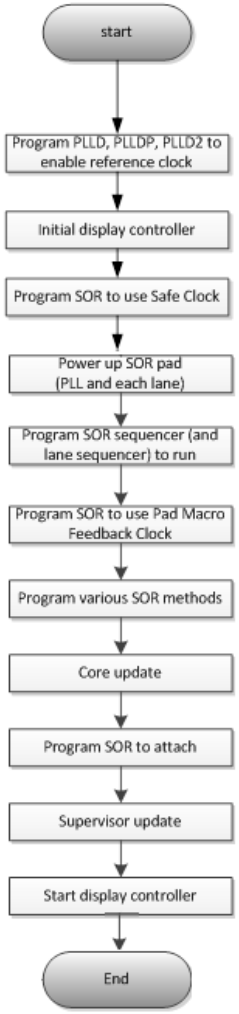
The overall sequence to start up HDMI is:

- Program power, pinmux, clocks, resets, etc.
- Copy HDCP keys from KFUSE to HDMI.
- Program display timing registers, etc.
- Program HDMI registers and SOR sequencer.
- Start HDMI SOR.

- Start display.

Since audio is entirely asynchronous to HDMI or the display, the HDA module can be started any time.

27.7.2.1 SOR Start



27.7.2.2 HDMI Start Sequence

Initialize Display Controller

Before programming the SOR, the display controller must be initialized.

```
STATE_ACCESS.WRITE_MUX ACTIVE
```

Set display to use the active copy of shadow register; therefore any programming to display register will be active immediately.

```
DISP_WIN_OPTIONS.SOR1_ENABLE ENABLE
```

This register controls the sor_enable signal in display which controls the enabling of display2sor1 clock gating and raster timing signals.

Program display pixel clock so dcpclk is valid

Program display window selection registers and window initial properties (can be overridden later)

The Display pixel clock should be programmed to be valid at this stage. The pixel clock feeds to the SOR pad CLOCKIN, which generates the CLOCKOUT, which will be used later by SOR as orclk in TMDS mode. Before SOR switches to use pad CLOCKOUT as orclk, the CLOCKOUT should be stable.

```
WIN_OPTIONS.WIN_ENABLE ENABLE
```

Enables display window.

Program Display Controller to Generate the Preamble as Expected by HDMI

Enforce display to send preamble signal as required by HDMI:

```
DISP_WIN_OPTIONS.SOR1_TIMING_CYA HDMI
```

Program HPULSE2 to Generate the Timing Signals for HDMI

Enforce display to send preamble signal as required by HDMI

```
DISP_SIGNAL_OPTIONS0.H_PULSE2_ENABLE ENABLE
H_PULSE2_CONTROL.H_PULSE2_MODE NORMAL
H_PULSE2_CONTROL.H_PULSE2_POLARITY HIGH
H_PULSE2_CONTROL.H_PULSE2_V_QUAL VACTIVE
H_PULSE2_CONTROL.H_PULSE2_LAST_END END_A
H_PULSE2_POSITION.H_PULSE2_START_A H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH - 10
H_PULSE2_POSITION.H_PULSE2_END_A H_PULSE2_START_A + 8
```

Miscellaneous registers required:

```
DISP_TIMING_OPTIONS.VSYNC_H_POSITION 1
DISP_CLOCK_CONTROL.PIXEL_CLK_DIVIDER PCD1
DISP_COLOR_CONTROL.DITHER_CONTROL    DISABLE
```

- Dither

  Dithering should be disabled, since HDMI uses full 8 bits per component.

- Component Range
- For most HDMI resolutions (i.e., everything except VGA 640x480), R/G/B should be scaled to lie in the [16, 235] nominal range; see the HDMI specification, Section 6.6. The range scaling is recommended to be done in CSC2.

In the CSC2 block:

```
CSC2_CONTROL.OUTPUT_COLOR_SELECT = RGB
CSC2_CONTROL.LIMIT_RGB_COLOR = ENABLE
```

For most HDMI modes, pixel values 0 and 255 must be removed. HDMI's NV_PDISP_INPUT_CONTROL.ARM_VIDEO_RANGE = LIMITED enables this.

Gamma

Most content uses sRGB gamma which is not quite the same as Rec. 709 gamma used by HDMI for most modes. Post-composite gamma can be changed in CMU.

Program SOR to Use Safe Clock and Program refclk

In system level, it is required to program CAR for SOR to use safe clock (default).

```
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL1 0
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL0 0
```

Program SOR reference clock, based on HDMI pixel clock – see NV_PDISP_SOR_REFCLK in spec. For example,

```
dispclk_div_8_2 = int(dispclk_freq / 1000000.0 * 4)
NV_PDISP_SOR_REFCLK.DIV_INT = dispclk_div_8_2 >> 2
NV_PDISP_SOR_REFCLK.DIV_FRAC = dispclk_div_8_2 & 0x3
```

Power up SOR Pad

- Deassert E_DPD which is controlled by PMC module.
- Wait for at least 5 µs.
- Deassert PDBG.
- Wait for at least 20 µs.
- Deassert PDPLL, PLLVCOPD and PLLCAPPD.
- Enable SOR PLL.
- Wait for at least 200 µs to PLL lock.

1. Wait for 5 µs
2. Deassert PDBG SOR.NV_PDISP_SOR_PLL2.AUX6 BANDGAP_POWERDOWN_DISABLE
3. Wait for 20 µs
4. Deassert PLLVDD, VCOPD, and PLLCAPPD

```
SOR.NV_PDISP_SOR_PLL3.PLLVDD_MODE V3_3
SOR.NV_PDISP_SOR_PLL0.PWRON
SOR.NV_PDISP_SOR_PLL0.VCOPDRESCIND
SOR.NV_PDISP_SOR_PLL2.AUX8 SEQ_PLLCAPPD_ENFORCE_DISABLE //Force PLLCAPPD to be asserted
```

5. Wait for 200 µs

```
SOR.NV_PDISP_SOR_PLL2.AUX7 PORT_POWERDOWN_DISABLE
SOR.NV_PDISP_SOR_PLL2.AUX2 OVERRIDE_POWERDOWN
```

6. Wait for 20 µs
7. Enable power to the lane

```
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_0  NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_1  NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_2  NO
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_3  NO
```

8. Program SOR lane sequencer

```
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.DELAY             15 //15us between each lane's power state change
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.NEW_POWER_STATE   PU //power up when triggers
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SEQUENCE          DOWN
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SETTING_NEW       TRIGGER
```

9. Then can poll SOR.NV_PDISP_SOR_LANE_SEQ_CTL for a DONE status

CHECK TMDS Macro Programming

This field should only be changed when MODE_BYPASS is set to DP_SAFE or when the SOR is in safe mode.

```
If (pclk_freq < 340MHz)
SOR.NV_PDISP_SOR_CLK_CNTRL.DP_LINK_SPEED G2_7
Else
SOR.NV_PDISP_SOR_CLK_CNTRL.DP_LINK_SPEED G5_4

SOR.NV_PDISP_SOR_CLK_CNTRL.DP_CLK_SEL  SINGLE_PCLK
```

Program SOR Sequencer

Once loadv arrives, SOR sequencer can be kicked off.

Enable sequencer

```
SOR.NV_PDISP_SOR_DP_SPARE0.SEQ_ENABLE YES
SOR.NV_PDISP_SOR_SEQ_CTL.PU_PC 0
SOR.NV_PDISP_SOR_SEQ_CTL.PU_PC_ALT 0
SOR.NV_PDISP_SOR_SEQ_CTL.PD_PC 8
SOR.NV_PDISP_SOR_SEQ_CTL.PD_PC_ALT 8
SOR.NV_PDISP_SOR_SEQ_CTL.SWITCH 0
```

Sequencer settings for HDMI

```
For ip=0 and 8,

SOR.NV_PDISP_SOR_SEQ_INST${ip}.WAIT_TIME 1
SOR.NV_PDISP_SOR_SEQ_INST${ip}.WAIT_UNITS VSYNC
SOR.NV_PDISP_SOR_SEQ_INST${ip}.HALT TRUE
SOR.NV_PDISP_SOR_SEQ_INST${ip}.PIN_A LOW
SOR.NV_PDISP_SOR_SEQ_INST${ip}.PIN_B LOW
SOR.NV_PDISP_SOR_SEQ_INST${ip}.DRIVE_PWM_OUT_LO TRUE
SOR.NV_PDISP_SOR_SEQ_INST${ip}.TRISTATE_IOS ENABLE_PINS
SOR.NV_PDISP_SOR_SEQ_INST${ip}.BLACK_DATA NORMAL
SOR.NV_PDISP_SOR_SEQ_INST${ip}.BLANK_DE NORMAL
SOR.NV_PDISP_SOR_SEQ_INST${ip}.BLANK_H NORMAL
SOR.NV_PDISP_SOR_SEQ_INST${ip}.BLANK_V NORMAL
SOR.NV_PDISP_SOR_SEQ_INST${ip}.ASSERT_PLL_RESET NORMAL
SOR.NV_PDISP_SOR_SEQ_INST${ip}.POWERDOWN_MACRO NORMAL
```

Program SOR to Use Pad Macro Feedback Clock

When the previous programming is done, need to switch sor clock to connect to feedback clock.

This switching is done by using command “sor_clk_sel pad” to enable in top_peatrans.

```
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SRC PLLD
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL1 1
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL0 1
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_DIVISOR
```

Program Other SOR Methods

Program general SOR setting related to head connected, hsync polarity, vsync polarity, de polarity, crc mode, bpp, display head control registers, display raster timing information.

To make display controller A as the primary driver for HDMI

To make display controller B as the primary driver for HDMI

```
SOR.NV_PDISP_INPUT_CONTROL.HDMI_SRC_SELECT DISPLAYB
SOR.NV_PDISP_INPUT_CONTROL.HDMI_SRC_SELECT DISPLAY (or) DISPLAYB
SOR.NV_PDISP_SOR_STATE1.ASY_PROTOCOL LVDS_CUSTOM //default value
```

Set asy_vsyncpol, asy_hsyncpol and asy_de according to panel requirements

```
if {0 == [string compare "positive" $hsync_polarity]} {
SOR.NV_PDISP_SOR_STATE1.ASY_HSYNCPOL POSITIVE_TRUE
} else {
```

```
SOR.NV_PDISP_SOR_STATE1.ASY_HSYNCPOL NEGATIVE_TRUE
}
if {0 == [string compare "positive" $vsync_polarity]} {
SOR.NV_PDISP_SOR_STATE1.ASY_VSYNCPOL POSITIVE_TRUE
} else {
SOR.NV_PDISP_SOR_STATE1.ASY_VSYNCPOL NEGATIVE_TRUE
}
if {0 == [string compare "positive" $DE_polarity]} {
SOR.NV_PDISP_SOR_STATE1.ASY_DEPOL POSITIVE_TRUE
} else {
SOR.NV_PDISP_SOR_STATE1.ASY_DEPOL NEGATIVE_TRUE
}
```

Set pixel depth. Only valid for HDMI is BPP_24_444 (5).

```
SOR.NV_PDISP_SOR_STATE1.ASY_PIXELDEPTH BPP_24_444
```

Program display head control registers

Set the dynamic range for the output. Display Port has a restriction that BPP_18_444 RGB mode must use VESA.

```
if {0 == [string compare "vesa" $dynamic_range]} {
SOR.NV_PDISP_HEAD_STATE0.DYNRANGE $head VESA
} else {
SOR.NV_PDISP_HEAD_STATE0.DYNRANGE $head CEA
}
SOR.NV_PDISP_HEAD_STATE0.COLORSPACE $head RGB //Only RGB is supported
```

Program Preamble timing in SOR

Enables SOR to use preamble signal generated by display:

```
SOR.NV_PDISP_SOR_DP_SPARE0.DISP_VIDEO_PREAMBLE_CYA ENABLE
```

Program display head control registers for raster timing

```
SOR.NV_PDISP_HEAD_STATE1.VTOTAL $head $total_height
SOR.NV_PDISP_HEAD_STATE1.HTOTAL $head $total_width
//sync_endy == v_ref_to_sync + v_sync_width
//constraint: sync_endy >= 1 (due to v_ref_to_sync >= 1)
SOR.NV_PDISP_HEAD_STATE2.VSYNC_END $head $sync_endy
//sync_endx == h_ref_to_sync + h_sync_width
SOR.NV_PDISP_HEAD_STATE2.HSYNC_END $head $sync_endx
//blank_endy == v_sync_width + v_backporch
SOR.NV_PDISP_HEAD_STATE3.VBLANK_END $head $blank_endy
//blank_endy == h_sync_width + h_backporch
SOR.NV_PDISP_HEAD_STATE3.HBLANK_END $head $blank_endx
//blank_starty == v_sync_width + v_back_porch + v_active (surface_height)
SOR.NV_PDISP_HEAD_STATE4.VBLANK_START $head $blank_starty
//blank_starty == h_sync_width + h_back_porch + h_active (surface_width)
SOR.NV_PDISP_HEAD_STATE4.HBLANK_START $head $blank_startx
```

Validate raster timing for display to meet Tegra X1 display constraints

Tegra X1 display Constraint 1: H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH > 20.

If the raster config violates this constraint, will adjust htotal, hblank_end and hblank_start values to meet the constraint as: H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH = 20.

```
if {$blank_endx + [REF_TO_SYNC.H_REF_TO_SYNC] < 20} {
            SOR.NV_PDISP_HEAD_STATE1.HTOTAL $head [expr 19 - [REF_TO_SYNC.H_REF_TO_SYNC] +
$total_width - $blank_endx]
            SOR.NV_PDISP_HEAD_STATE3.HBLANK_END $head [expr 19 - [REF_TO_SYNC.H_REF_TO_SYNC]]
            SOR.NV_PDISP_HEAD_STATE4.HBLANK_START $head [expr 19 - [REF_TO_SYNC.H_REF_TO_SYNC] -
$blank_endx + $blank_startx]
}
```

Attach SOR

1. Program head owner

```
if { use displayA } {
SOR.NV_PDISP_SOR_STATE1.ASY_OWNER HEAD0
} else {
      SOR.NV_PDISP_SOR_STATE1.ASY_OWNER HEAD1
}
SOR.NV_PDISP_SOR_STATE1.ASY_SUBOWNER BOTH
```

2. Core update

```
sor_update
proc sor_update {} {
    SOR.NV_PDISP_SOR_STATE0.UPDATE 0
    flush
    SOR.NV_PDISP_SOR_STATE0.UPDATE 1
    flush
    SOR.NV_PDISP_SOR_STATE0.UPDATE 0
    flush
}
```

3. Attach SOR

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE    SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED      YES
sor_super_update
proc sor_super_update {} {
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
    flush
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 1
    flush
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
    flush
}
```

4. Here can poll NV_PDISP_SOR_TEST for attach status (SOR.NV_PDISP_SOR_TEST.ATTACHED.TRUE)
5. Set OR mode to NORMAL

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODENORMAL
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
sor_super_update
```

6. Head awake

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE AWAKE
```

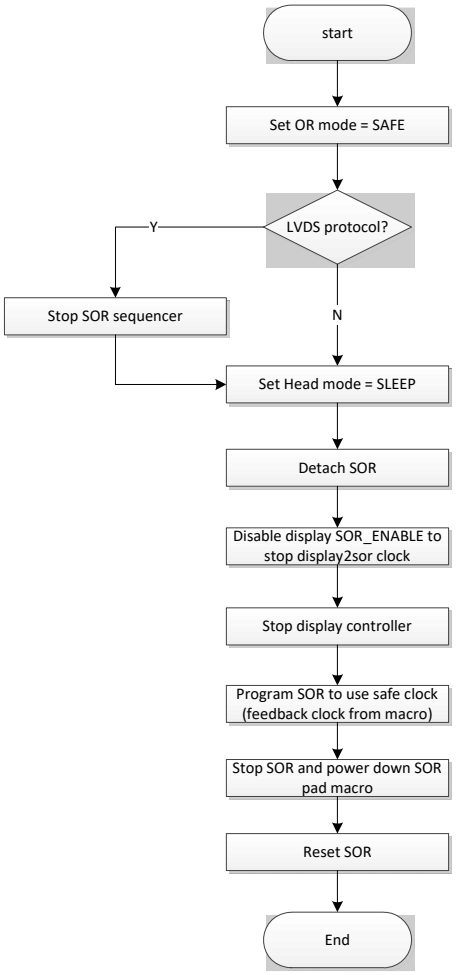
```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      NORMAL
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
```

Start the Display Controller

Now start the display controller by programming the display registers.

27.7.2.3 SOR Stop

General sequence



27.7.2.4 HDMI Stop Sequence

Set OR mode to SAFE

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE AWAKE
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
sor_super_update
proc sor_super_update {} {
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
    flush
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 1
    flush
    SOR.NV_PDISP_SOR_SUPER_STATE0.UPDATE 0
```

```
    flush
}
```

Here can poll OR mode for SAFE (SOR.NV_PDISP_SOR_PWR.MODE.SAFE)

Set Head mode to SLEEP

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        YES
sor_super_update
```

Detach SOR

```
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_HEAD_OPMODE SLEEP
SOR.NV_PDISP_SOR_SUPER_STATE1.ASY_ORMODE      SAFE
SOR.NV_PDISP_SOR_SUPER_STATE1.ATTACHED        NO
sor_super_update
```

Here can poll SOR TEST to detach (SOR.NV_PDISP_SOR_TEST.ATTACHED.FALSE)

Core update: promote SOR_SET_CONTROL to arm state

```
SOR.NV_PDISP_SOR_STATE1.ASY_OWNER NONE
SOR.NV_PDISP_SOR_STATE1.ASY_SUBOWNER NONE
SOR.NV_PDISP_SOR_STATE1.ASY_PROTOCOL LVDS_CUSTOM
sor_update
proc sor_update {} {
    SOR.NV_PDISP_SOR_STATE0.UPDATE 0
    flush
    SOR.NV_PDISP_SOR_STATE0.UPDATE 1
    flush
    SOR.NV_PDISP_SOR_STATE0.UPDATE 0
    flush
}
```

Disable Display SOR_ENABLE to Stop display2sor Clock

Set display to use the active copy of shadow register, this enables the disabling of PW* registers take effect immediately (otherwise it will wait till the frame end since display uses the assembly copy of registers)

```
STATE_ACCESS.WRITE_MUX ACTIVE
DISP_WIN_OPTIONS.SOR_ENABLE DISABLE //this will stop display2sor pixel clock
```

Stop the Display Controller

```
DISPLAY_COMMAND.DISPLAY_CTRL_MODE STOP // stop sending frame at the next frame boundary
DISPLAY_POWER_CONTROL.PW0_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW1_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW2_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW3_ENABLE DISABLE
DISPLAY_POWER_CONTROL.PW4_ENABLE DISABLE
```

Program SOR to switch from Feedback Clock to Safeclk

Requires CAR to be programmed for SOR to use safeclk.

```
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL1 0
CLK_RST_CONTROLLER_CLK_SOURCE_SOR1.SOR1_CLK_SEL0 0
```

Stop SOR and Power Down SOR Pad Macro

- Power down each lane of SOR.
- Set PDPLL, PLLVCOPD, PLLCAPPD to 1.
- Wait for at least 5 µs.
- Power down PDBG: set PDBG to 1.
- Wait for at least 5 µs.
- Power down E_DPD: set E_DPD to 1.

1. Power down each lane

```
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_0 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_1 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_2 YES
SOR.NV_PDISP_SOR_DP_PADCTL0.PD_TXD_3 YES
```

2. Stop lane sequencer

```
SOR.NV_PDISP_SOR_LANE_SEQ_CTL -dirty //enables config with the same value as is
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.DELAY             15
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.NEW_POWER_STATE   PD
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SEQUENCE          DOWN
SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SETTING_NEW       TRIGGER
```

3. Here can poll SOR lane sequencer ctrl status for SETTING_NEW done (SOR.NV_PDISP_SOR_LANE_SEQ_CTL.SETTING_NEW.DONE)
4. Power down PDPORT to power down all the output links and lanes

```
SOR.NV_PDISP_SOR_PLL2.AUX7          PORT_POWERDOWN_ENABLE
```

5. Wait for 20 µs
6. Power down PDPLL, PLLVCOPD, PLLCAPPD

```
SOR.NV_PDISP_SOR_PLL0.PWR           OFF
SOR.NV_PDISP_SOR_PLL0.VCOPD         ASSERT
SOR.NV_PDISP_SOR_PLL2.AUX1          SEQ_PLLCAPPD_OVERRIDE
SOR.NV_PDISP_SOR_PLL2.AUX8          SEQ_PLLCAPPD_ENFORCE_ENABLE
```

7. Wait for 20 µs
8. Power down PDBG

```
SOR.NV_PDISP_SOR_PLL2.AUX6          BANDGAP_POWERDOWN_ENABLE
```

9. Set E_DPD to 1 (PWR)
10. Program E_DPD to 1 in PMC registers.

Reset SOR

Tegra X1 requires SOR to be reset after power down. Because after stop SOR, display's hvtgen does not clean the counters. Therefore if SOR is restarted, the counters will continue to count and result in a function error.

The reset is implemented by programming the following CAR register (Reset device SOR0):

```
CLK_RST_CONTROLLER_RST_DEVICES_X
```

27.7.3 Audio

27.7.3.1 HDA Audio Codec Initialization

HDA is the only audio source, the codec must be initialized before audio can flow. The internal codec uses industry-standard verbs, so in theory, the out-of-box driver should correctly discover and configure the codec. See the HD Audio specification. Below is a summary of the necessary verbs:

1. AOC widget: SET_CONV_STREAM_CHAN with desired output stream ID
2. AOC widget: SET_CONV_FMT to desired audio format (sampling frequency, sample depth, number of channels)
3. AOC widget: SET_DIGITAL_CONV_CONTROL1
   - Bit 0: 1 (DigEn=1)
   - Bit 3..6: channel status pre/copy/audio/pro bits
   - Bit 7: channel status CC bit 7 (called GEN_LEVEL)
4. AOC widget: SET_DIGITAL_CONV_CONTROL2:
   - Bit 6:0: channel status CC bits 6:0
5. Pin widget: SET_CONVERTER_CHANNEL_TO_DIGITAL_SLOT_MAPPING= 0x22 and SET_CONVERTER_CHANNEL_TO_DIGITAL_SLOT_MAPPING= 0x33 to undo default swap of channels 2 and 3
6. AOC widget: SET_STRIPE_CONTROL to select number of lanes. Recommend programming to 2 lanes all the time.
7. Pin widget: SET_PIN_WIDGET_CTRL = 0x40, to enable output and use PCM packets.
8. If desired to use HDA for info packets, program them with SET_DIP_INDEX/SET_DIP_DATA and enable with SET_DIP_XMIT_CTRL.

27.7.3.2 HDMI Audio Settings

Two methods of generating the CTS portion of ACR packets are supported:

- In “Force SW CTS”, software computes CTS for the current audio and pixel clocks and writes it to registers.
- In “Hardware CTS”, hardware computes CTS automatically.

Likewise, two methods for generating N portion are supported.

- In “SW N”, software programs current desired N into 0441_SUBPACK_HIGH register and AUDIO_N.VALUE.
- In “HW N”, software programs all the ACR_*_SUBPACK_HIGH per-frequency N, then hardware chooses the one to use.

1. Set NV_PDISP_SOR_HDMI_ACR_CTRL (all fields) to 0
2. Program LOOKUP per table.
3. Based on pixel clock and audio sample clock frequencies, program the appropriate ACR N value into NV_PDISP_SOR_HDMI_ACR_0441_SUBPACK_HIGH (see HDMI 1.2a specification, Section 7.2.3). LSB of N goes in SB6, and MSB into SB4.
4. Likewise, program the appropriate ACR CTS value into NV_PDISP_SOR_HDMI_ACR_0441_SUBPACK_LOW. The LSB of CTS goes in SB3, and the MSB into SB1.
5. Program the computed AVAL into the appropriate SOR_AUDIO_AVAL_ register
6. Set NV_PDISP_SOR_HDMI_ACR_*_SUBPACK_HIGH.ENABLE = YES.
7. Set NV_PDISP_SOR_HDMI_SPARE = 0x10000 OR’d with bits 0 and 1
8. Program the audio source with SOR_AUDIO_CNTRL0.SOURCE_SELECT = HDAL for HDA, or AUTO. AUTO automatically selects HDA if HDMI’s internal HDA codec has been enumerated by HDA controller.
9. Enable HDMI audio: Set NV_PDISP_SOR_HDMI_CTRL.ENABLE = EN, NV_PDISP_SOR_HDMI_GENERIC_CTRL.AUDIO = EN, AUDIO_ENGINE.PWRDWN=0

Note: *CTS and N values depend on the ratio of the actual pixel clock and audio clock rates. The chip's clock generators cannot create every frequency exactly. For example 25.2 MHz (for 640x480p@60) cannot be exactly generated – 25.25 is the result. Thus CTS and N must be adjusted accordingly.*

27.7.4 Audio Codec

The HDMI block in Tegra X1 contains an audio codec that is connected to the HDA controller on the chip. The audio codec identifies as: 0x10de0029. Note that the audio codec is used in both DP and HDMI. The table below contains a list of all supported verbs for this codec:

Table 163: Supported Verbs for Audio Codec

Node	Verb (Verb ID)	Payload	Response (32 bits)	Default Power on Value/Notes
Root	Get Parameter (0xF00)	Parameter ID	Parameter Value	NA
Audio Function Group	Get Parameter (0xF00)	Parameter ID	Parameter Value	NA
	Get Power State (0xF05)	0	Bits [31:11] are 0 Bit [10] = PS-Settings Reset Bit [9] = PS-ClkStopOk Bit [8] = PS-Error Bits [7:4] = PS-Act Bits [3:0] = PS - Set	PS-Settings Reset: 1 PS-ClkStopOk: 0x1 PS-Error: 0x0 PS-Act: 0x3 (D3hot) PS-Set: 0x3 (D3hot) D3hot is chosen as the default power on state in accordance with Intel ECR HDA015-B
	Set Power State (0x705)	Bits [7:4] are 0 and Bits [3:0] = PS – Set Value	0	
	Get Sub-System ID (0xF20)	0	Bits [31:16] = Vendor ID Bit [15:8] = Board SKU Bits [7:0] = Assembly ID	Vendor ID: 0x10DE Board SKU: 0x01 Assembly ID: 0x01
	Set Sub-System ID -1 (0x720)	Bits [7:0] of subsystem ID	0	
	Set Sub-System ID -2 (0x720)	Bits [15:8] of subsystem ID	0	
	Set Sub-System ID -3 (0x720)	Bits [23:16] of subsystem ID	0	
	Set Sub-System ID -4 (0x720)	Bits [31:24] of subsystem ID	0	
	Reset	0	0	In accordance with Intel ECN, treated as single AFG reset.
	Set Scratch Reg0, Byte0 (0x FA7) Vendor Defined	Bits [7:0] transparent to hardware	0	Reg0, Byte0: 0x00 Reg0, Byte1: 0x00 Reg0, Byte2: 0x00 Reg0, Byte3: 0x00
	Set Scratch Reg0, Byte1 (0xFA8) Vendor Defined	Bits [7:0] transparent to hardware	0	
	Set Scratch Reg0, Byte2 (0xFA9) Vendor Defined	Bits [7:0] transparent to hardware	0	
	Set Scratch Reg0, Byte3 (0xFAA) Vendor Defined	Bits [7:0] transparent to hardware	0	
	Get Scratch Reg0 (0xFA6) Vendor Defined	0	Bits [31:24]=Reg0, Byte3 Bits [23:16]=Reg0, Byte2 Bits [15:8]=Reg0, Byte1 Bits [7:0]=Reg0, Byte0	

Table 163: Supported Verbs for Audio Codec

Node	Verb (Verb ID)	Payload	Response (32 bits)	Default Power on Value/Notes
Audio Function Group	Set Scratch Reg1, Byte0 (0xFAC) Vendor Defined	Bits [7:0] transparent to hardware	0	Reg1, Byte0: 0x00 Reg1, Byte1: 0x00 Reg1, Byte2: 0x00 Reg1, Byte3: 0x00
	Set Scratch Reg1, Byte1 (0xFAD) Vendor Defined	Bits [7:0] transparent to hardware	0	
	Set Scratch Reg1, Byte2 (0xFAE) Vendor Defined	Bits [7:0] transparent to hardware	0	
	Set Scratch Reg1, Byte3 (0xFAF) Vendor Defined	Bits [7:0] transparent to hardware	0	
	Get Scratch Reg1 (0xFAB) Vendor Defined	0	Bits [31:24]=Reg1, Byte3 Bits [23:16]=Reg1, Byte2 Bits [15:8]=Reg1, Byte1 Bits [7:0]=Reg1, Byte0	
	Get Display Scratch Registers (0xFA5)	Bits [7:2] are 0 Bits [1:0] specify one of the 4 registers	Bits [31:0] = Reg3/2/1/0 (index = 3/2/1/0)	Reg3-0: 0x00000000 (however, they are reset by display reset)
Output Converter	Get Parameter (0xF00)	Parameter ID	Parameter Value	NA
	Get Converter Format (0x0A)	0	Bits [31:16] are 0 Bit [15] = Stream Type Bit [14] = Base Rate Bits [13:11] = Base Multiplier Bits [10:8] = Base Divisor Bit [7] = 0 Bits [6:4] = Bits per sample Bits [3:0] = number of channels	Stream Type: 0x0 (PCM) Base Rate: 0x0 (48 KHz) Base Multiplier: 0x0 Base Divisor: 0x0 Bits per sample: 0x1 (16 bits) Number of channels: 0x1 (2 channels)
	Set Converter Format (0x02)	Bits [31:16] are 0 and bits [15:0] contain the format	0	
	Get Converter Control (0xF0D)	0	Bits [31:24] are 0 Bit [23] = Keep Alive Enable Bits [22:20] are 0 Bits [19:16] = IEC Coding Type Bits [14:8] = CC Bit [7] = L Bit [6] = PRO Bit [5] = /AUDIO Bit [4] = COPY Bit [3] = PRE Bit [2] = VCFG Bit [1] = V Bit [0] = DigEn	Keep alive enable: 0x 0 IEC Coding Type: 0x 0 CC: 0x00 L: 0x0 PRO: 0x0 /AUDIO: 0x0 COPY: 0x0 PRE: 0x0 VCFG: 0x0 V: 0x0 DigEn: 0x0
	Set Converter Control-1 (0x70D)	Bits [7:0] of SIC (SPDIF IEC Control)	0	
	Set Converter Control-2 (0x70E)	Bits [15:8] of SIC	0	

Table 163: Supported Verbs for Audio Codec

Node	Verb (Verb ID)	Payload	Response (32 bits)	Default Power on Value/Notes
Output Converter	Get Stream, Channel (0xF06)	0	Bits [31:8] are 0 Bits [7:4] = stream number Bits [3:0] = Lowest channel number used by converter (0-indexed)	Stream #: 0 Channel #: 0
	Set Stream, Channel (0x706)	Bits [7:4] are stream number Bits [3:0] are channel	0	
	Get Unsolicited Response (0xF08)	0	Bits [31:8] are 0 Bit [7] = Enable Bit [6] is 0 Bits [5:0] = Tag	Enable: 0x0 Tag: 0x00
	Set Unsolicited Response (0x708)	Bit [7] is Enable Bit [6] is 0 Bits [5:0] Tag	0	
	Get Stripe Control (0xF24)	0	Bits [31:23] are 0 Bits [22:20] = Stripe capability Bits [7:2] are 0 Bits [1:0] = Stripe Control	Stripe capability: 0x7 Stripe Control: 0x0
	Set Stripe Control (0x724)	Bits [31:3] are 0 Bits [2:0] set stripe control	0	
	Get Converter Channel Count (0xF2D)	0	Bits [31:8] are 0 Bits [7:0] = Channel count (0-indexed)	Channel Count: 0x0
	Set Converter Channel Count (0x72D)	Bits [7:0] set channel count	0	
Pin Widget	Get Parameter (0xF00)	Parameter ID	Parameter Value	NA
	Get Connection List Entry (0xF02)	Bits [7:0] are 0	Bits [31:8] are 0 Bits [7:0] = 0x4	NA
	Get Pin Widget Control (0xF07)	0	Bits [31:8] are 0 Bit [7] = H-Phn Enable Bit [6] = Out Enable Bit [5] = In Enable Bits [4:3] are 0 Bit [2] = VRefEn Bits [1:0] = EPT	H-Phn Enable (NA): 0x0 Out Enable: 0x0 In Enable (NA): 0x0 VRefEn (NA): 0x0 EPT: 0x0 EPT is defined in Intel ECN HDA035
	Set Pin Widget Control (0x707)	Bits [7] and Bits [5:2] are 0x00 Bit [6] specifies Out Enable Bits [1:0] specifies EPT	0	
	Get Unsolicited Response (0xF08)	0	Bit [7] = Enable Bit [6] are 0 Bits [5:0] = Tag	Enable: 0x0 Tag: 0x00
	Set Unsolicited Response (0x708)	Bits [7] is Enable Bit [6] is 0 Bits [5:0] Tag	0	
	Get Pin Sense (0xF09)	0	Bit [31] = Presence Detect Bit [30] = ELDV Bits [29:0] are 0	PD: 0x0 ELDV: 0x0

Table 163: Supported Verbs for Audio Codec

Node	Verb (Verb ID)	Payload	Response (32 bits)	Default Power on Value/Notes
Pin Widget	Get Configuration Defaults (0xF1C)	0	Bits [31:30] = Port Connectivity Bits [29:24] = Location Bits [23:20] = Default Device Bits [19:16] = Connection Type Bits [15:12] = Color Bits [11:8] = Misc Bits [7:4] = Default Association Bits [3:0] = Sequence	Port Connectivity: 0x0 Location: 0x18 (HDMI or DP) Default Device: 0x5 (Digital other out) Connection Type: 0x6 (other digital) Color: 0x0 Misc: 0x0 Default association: 0x1 Sequence: 0x0
	Set Configuration Defaults-1 (0x71C)	Bits[7:4] are Default Association and Bits [3:0] are Sequence	0	
	Set Configuration Defaults-2 (0x71D)	Bits[7:4] are Color and Bits [3:0] are Misc	0	
	Set Configuration Defaults-3 (0x71E)	Bits[7:4] are Connection Type and Bits [3:0] are Default Device	0	
	Set Configuration Defaults-4 (0x71F)	Bits[7:4] are Location and Bits [3:0] are Port Connectivity	0	
	Get DIP Size Info	Bits [7:4] are 0 Bits [3] is set for ELD Buffer Bits [2:0] specifies packet Index for DIP Buffer (NA if bit [3] is 1)	Bits [31:8] are 0 Bits [7:0] = Size of either ELD buffer or DIP buffer (0-indexed)	NA
	Get ELD Data (0xF2F)	Bits [7:0] are offset into ELD memory	Bit [31] = Byte at specified offset is valid Bits [30:8] are zero Bits [7:0] = ELD byte	NA
	Get HDMI DIP Index (0xF30)	0	Bits [7:5] = Current Packet Index Bits [4:0] = Byte Index in this packet	Packet Index: 0x0 Byte Index: 0x00
	Set HDMI DIP Index (0x730)	Bits [7:5] specify value of PI Bits [4:0] specify the byte index	0	
	Get HDMI DIP Data (0xF31)	0	Bits [31:8] are 0 Bits [7:0] = Byte as specified by PI and byte index	NA
	Set HDMI DIP Data (0x731)	Bits [7:0] is byte to be written at byte index of PI	0	
	Get HDMI DIP Transmit Control (0xF32)	0	Bits [31:8] are 0 Bits [7:6] = Transmit control for PI Bits [5:0] are 0	Transmit control: 0x0
	Set Transmit Control (0x732)	Bits [7:6] specify value for transmit control Bits [5:0] are 0	0	

Table 163: Supported Verbs for Audio Codec

Node	Verb (Verb ID)	Payload	Response (32 bits)	Default Power on Value/Notes
Pin Widget	Get CP_Control (0xF33)	0	Bits [31:10] are 0 Bit [9] = CES Bit [8] = READY Bits [7:3] = Sub-Tag Bit [2]= 0 Bits [1:0] = Requested CP state	READY: 0x0 Sub-Tag: 0x00 Requested CP state: 0x0
	Set CP_Control (0x733)	Bits [7:3] specify the sub-tag Bit [2] is 0 Bits[1:0] specify the CP state	0	
	Get ASP Channel Mapping (0xF34)	Bits [7:4] are 0 Bits [3:0] is ASP slot number	Bits [31:8] are 0 Bits [7:4] = Converter channel number Bits [3:0] = Slot number	ASP slot 0: 0 ASP slot 1: 1 ASP slot 2: 3 ASP slot 3: 2 ASP slot 4: 4 ASP slot 5: 5 ASP slot 6: 6 ASP slot 7: 7
	Set ASP Channel Mapping (0x734)	Bits [7:4] specify the converter channel number Bits [3:0] specify the ASP slot number	0	

For interoperability between HDMI and HDMI codec drivers, the HDMI codec contains a pair of scratch registers that are reflected in the HDMI block's SOR_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH0/1_0 registers. The HDMI codec driver can access these scratch registers using the following HDA verbs:

```
Get Scratch Reg0 (0xFA6)
Set Scratch Reg0, Byte 0 (0xFA7)
Set Scratch Reg0, Byte 1 (0xFA8)
Set Scratch Reg0, Byte 2 (0xFA9)
Set Scratch Reg0, Byte 3 (0xFAA)

Get Scratch Reg1 (0xFAB)
Set Scratch Reg1, Byte 0 (0xFAC)
Set Scratch Reg1, Byte 1 (0xFAD)
Set Scratch Reg1, Byte 2 (0xFAE)
Set Scratch Reg1, Byte 3 (0xFAF)
```

The content of these registers is defined entirely by software; hardware doesn't interpret it in any way. The only exception is that an interrupt is raised in the HDMI block whenever the most significant bit of either scratch register changes. The interrupt is reflected in the SOR_NV_PDISP_INT_STATUS_0 register's CODEC_SCRATCH0/1 bits.

A similar mechanism exists for communicating data from the HDMI driver to the HDMI codec driver. The SOR_NV_PDISP_SOR_AUDIO_HDA_SCRATCH0/1/2/3_0 registers in the HDMI block are reflected in the HDMI codec's display scratch registers, which in turn can be accessed using the following HDA verbs:

```
Get Display Scratch Registers (0xFA5)
```

Changing the most significant bit of these registers causes an unsolicited response event to be emitted for the HDMI codec. Otherwise the content is defined entirely by software; hardware doesn't interpret it in any way.

27.7.5 Miscellaneous

For Audio to HDMI signaling, the HDA Codec has SCRATCH verbs that are reflected in the HDMI SOR_AUDIO_HDA_CODEC_SCRATCH* registers and CODEC_SCRATCH* interrupts.

For HDMI to Audio signaling, HDMI SOR_AUDIO_HDA_SCRATCH* registers, when written, are reflected to the HDA Codec GET_NV_SCRATCH_REGN_FROM_DISP verb and cause an Unsolicited Response (interrupt in the HDA Controller).

27.7.6 Other HDMI Configurations

1. Set NV_PDISP_SOR_HDMI_CTRL.MAX_AC_PACKET based on horizontal blanking width. Using display timing registers, MAX_AC_PACKET = floor((H_SYNC_WIDTH + H_BACK_PORCH + H_FRONT_PORCH – HDMI_CTRL.REKEY – 18) / 32)
2. Program and enable Audio InfoFrames. When using HDA audio, its contents come from verbs sent to the internal codec.
3. Program and enable AVI InfoFrames. When using HDA Audio, this *may* come from the HDA Codec if desired, using one of its Generic packets. If so, do not enable the HDMI one:
   - Set PDISP_SOR_HDMI_AVI_INFOFRAME_HEADER = 0x000D0282
   - Program checksum, PB1, PB2, PB3 in HDMI_AVI_INFOFRAME_SUBPACK0_LOW. PB4 contains the video format code from the CEA-861-D specification. See HDMI and EIA/CEA-861-D for all the fields. CRC is computed over header and PB1…PB13 as above.
   - Program PB4…PB6 in HDMI_AVI_INFOFRAME_SUBPACK0_HIGH, and PB7..PB13 in HDMI_AVI_INFOFRAME_SUBPACK1_LOW / HIGH.
   - Set HDMI_AVI_INFOFRAME_CTRL.ENABLE = EN
4. For HDMI 1.4 3D, program and enable the GENERIC InfoFrame to send an HDMI Vendor-Specific InfoFrame. When using HDA Audio, this *might* come from the HDA Codec, if desired, using one of its Generic packets. If so, do not enable the HDMI one.
   - Set NV_PDISP_SOR_HDMI_GENERIC_HEADER = 0x00LL0181 where LL is length of packet – 0x05 for the basic version described here.
   - HDMI_GENERIC_SUBPACK0_LOW.PB1/PB2/PB3 = 0x03/0c/00 (IEEE registration ID)
   - PB4 = 0x40 (Hdmi_video_format = 3d_structure present)
   - PB5 = 0x00 (3D_Structure = Frame packing)
   - PB6...PB27 = 0 (no 3D_Ext_Data for Frame packing)
   - Compute checksum over header and PB1..PB27 and put in PB0, as above.
   - HDMI_GENERIC_CTRL.ENABLE=EN. OTHER, SINGLE, HBLANK= DIS.
   - Be sure to keep HDMI_GENERIC_CTRL.AUDIO = EN

To support DVI rather than HDMI, set NV_PDISP_SOR_HDMI_CTRL.ENABLE = NO.

27.7.7 Display Raster Timing Register Programming

The Tegra X1 display has a legacy constraint of V_REF_TO_SYNC, which requires programming display register V_REF_TO_SYNC >=1. However, on the SOR side, the SOR design has an assumption of V_REF_TO_SYNC = 0. To resolve such a conflict, the DC raster timing programming in Tegra X1 is required to change when connected to SOR. This change affects software programming for DC raster timing on V_BACK_PORCH and V_FRONT_PORCH.

SOR and DC timing parameters have following constraints:

```
SOR.HSYNC_END    = DC.H_SYNC_WIDTH - 1
SOR.VSYNC_END    = DC.V_SYNC_WIDTH - 1
SOR.HBLANK_END   = SOR.HSYNC_END  + DC.H_BACK_PORCH
SOR.VBLANK_END   = SOR.VSYNC_END  + DC.V_BACK_PORCH + DC.V_REF_TO_SYNC SOR.HBLANK_START =
SOR.HBLANK_END + DC. H_DISP_ACTIVE
SOR.VBLANK_START = SOR.VBLANK_END + DC. V_DISP_ACTIVE
```

Because eDP configuration must match the panel specification and SOR design specification:

```
SOR.HBLANK_END   = h_sync_width + h_back_porch - 1;
SOR.HBLANK_START = h_sync_width + h_back_porch + h_active_width - 1;
SOR.VBLANK_END   = v_sync_width + v_back_porch - 1;
```

```
SOR.VBLANK_START = v_sync_width + v_back_porch + v_active_width - 1;
```

Note: *sync_width, back_porch, active_width, front_porch are params defined in the panel specification.*

So DC raster timing parameters should be modified as following:

```
DC.V_BACK_PORCH  = v_back_porch - DC.V_REF_TO_SYNC;
DC.V_FRONT_PORCH = v_front_porch + DC.V_REF_TO_SYNC;
```

Usually, DC.V_REF_TO_SYNC can be forced to 1.

27.7.8 HDA

The Tegra HDA controller has two parts: a bus interface block to the rest of the system named IPFS, and the actual HDA core controller.

The HDA core is implemented to an industry standard host controller specification published by Intel and known as ‘HD Audio’. This standard is sometimes also referred by its earlier name which was ‘Azalia’. For programming information on the HDA core, refer to http://www.intel.com/content/www/us/en/chipsets/high-definition-audio.html/

27.7.8.1 Clock and Reset

1. HDA controller needs two clock sources: hda_clk and hda2codec_2x_clk for the HDA core logic and the HDA link logic, respectively. The typical clock frequency for hda_clk is 108 MHz, but hda2codec_2x_clk should be set to 48 MHz, specified by the protocol spec. In fact, the protocol spec only designates the frequency of 1x clock, 24 MHz, but CAR generates the 2x clock and uses a clock divider to generate the 1x clock.
2. Set the clock source of both clocks through CAR programming. A typical way is to program PLLP to generate 216 MHz pllp_out0 and set the clock source of HDA_CLK/HDA2CODEC_2X_CLK to pllp_out0 with divider values 2 and 4.5, respectively.
3. Once the clocks are up and running, the reset signals should be de-asserted through programming HDA_RST and HDA2CODEC_2X_RST.

27.7.8.2 IPFS and FPCI

1. While the HDA controller is connected to the APB bus, it is wrapped with protocol conversion logic because it was also used in other NVIDIA chips. To be able to access the core configuration registers, some wrapper registers should be programmed accordingly. IPFS converts transactions between both the slave APB interface and the master MC protocol, mapping them to a bus protocol referred to as ‘FPCI’ which is the native interface of the HDA core. Also, the FPCI wrapper has PCI configuration registers.
2. IPFS is a flexible block designed to support multiple IPs which need various BAR addresses and sizes. However, in this instance, the HDA core logic needs only a 16 KB BAR0 region, which is already set in the IPFS registers. Software does not have to move the preset BAR0 region.
3. Simply turn on IPFS with programming to HDA_CONFIGURATION_0 with EN_FPCI, bit 0, enabled. The actual address of the register is NV_ADDRESS_MAP_HDA_BASE + HDA_CONFIGURATION_0.
4. The PCI configuration register can be accessed through using addresses starting from NV_ADDRESS_MAP_HDA_BASE + NV_HDA_APB_DFPCI_CFG_OFFSET. The first register, T_AZA_CFG_0 has read-only information such as vendor ID and device ID etc. To turn on the FPCI bus master, turn on T_AZA_CFG_1 with IO_SPACE, bit 0, MEM_SPACE, bit1 and BUS_MASTER, bit 2.

27.7.8.3 Pinmux

There are 4 pinmuxing options for the external codec: DAP1, DAP2, PE0 and PE2. Depending on the use case/application, software can choose one among the options. Refer to the pinmux table for the option details. The configuration registers are defined in the Pinmux Registers section of the Multi-purpose I/O Pins and Pin Multiplexing (Pinmuxing) chapter in this TRM. Make sure that the pinmuxes for all 5 external signals, SYNC, SDO, SDI, BCLK and RESET are programmed accordingly.

27.7.8.4 HDA Configuration

1. From cold reset, the HDA is still in the reset state. The first thing is to wake up HDA by de-asserting CRST. The CRST bit is in AZA_GCTL_0 whose address is NV_ADDRESS_MAP_HDA_BASE + NV_HDA_APB_BAR0_START + AZA_GCTL_0. Writing 1 to the bit wakes up HDA. Since resetting takes time, software should check the status, waiting for read back of CRST becoming 1.
2. Once CRST becomes 1, the HDA controller is ready to send commands (verbs) to codecs.
3. Before sending any real audio data to the internal codec, software may want to check if HDMI is properly woken up.
4. For ring buffer programming for command and data, refer to the HD Audio specification.

27.7.8.5 Other Configurations

1. When CPU gets an interrupt from HDA, it should read the interrupt status register in HDA to resolve any potential data coherency. This is required because the write transactions issued before the interrupt can be still in-flight. Reading the interrupt status register flushes all the write transactions before the register read value is returned. If there are many pending write transactions in FIFO, a simple read transaction can take some time due to the write transaction performance. The latency can be adjusted through the priority change in the MC register. See for the MC register spec for the details. This feature is turned on by default, but it can be defeated by setting HDA_CONFIGURATION_DFPCI_RSPPASSPW.
2. MC does not guarantee the original ordering of write transactions issued by HDA; especially if their addresses are more than 1 KB apart. By default, IPFS maintains the ordering of write transactions with the cost of performance. Usually, the write ordering is not critical unless some memory flags/status bits are used. Since HDA software is not expected to use such memory flags, it is recommended to turn on HDA_CONFIGURATION_UFPCI_PWPASSPW, which improves the write performance.

27.7.9 HDCP

27.7.9.1 Enable Encryption

Microcode handles most parts of the authentication sequence. Once the panel is authenticated and Session key is established, microcode enables hardware to start encryption. Enabling encryption should follow the below sequence:

Start State: _ENABLE = _NO; CRYPT_STS = _INACTIVE

1. Write NV_PDISP_SOR_HDCP22_LC128_MSB_0
2. Write NV_PDISP_SOR_HDCP22_LC128_LSB1_0
3. Write NV_PDISP_SOR_HDCP22_LC128_LSB2_0
4. Write NV_PDISP_SOR_HDCP22_LC128_LSB3_0 Write NV_PDISP_SOR_HDCP22_AES_CTR_KEY_LSB3
5. Write NV_PDISP_SOR_HDCP22_AES_CTR_KEY_LSB2
6. Write NV_PDISP_SOR_HDCP22_AES_CTR_KEY_LSB1
7. Write NV_PDISP_SOR_HDCP22_AES_CTR_KEY_MSB
8. Write NV_PDISP_SOR_HDCP22_AES_CTR_DATA_LSB
9. Write NV_PDISP_SOR_HDCP22_AES_CTR_DATA_MSB
10. Write NV_PDISP_SOR_HDCP22_CTRL_INIT_TRIGGER
11. Poll NV_PDISP_SOR_HDCP22_CTRL_INIT_DONE
12. Write NV_PDISP_SOR_HDCP22_CTRL_ENABLE_YES
13. Poll NV_PDISP_SOR_HDCP22_CTRL_CRYPT_STS = _ACTIVE

End State: _ENABLE = _YES; CRYPT_STS = _ACTIVE

27.7.9.2 Toggling Encryption

Start State: _ENABLE = _YES; CRYPT_STS = _ACTIVE

1. Write _ENABLE = _NO
2. Optional: Software waits for _CRYPT_STS to indicate _INACTIVE
3. Sometime later Write _ENABLE = _YES

End State: _ENABLE = _YES; CRYPT_STS = _ACTIVE

Redo Authentication

Start State: _ENABLE = _YES; CRYPT_STS = _ACTIVE

1. Write _ENABLE = _NO
2. Poll CRYPT_STS = INACTIVE
3. Session Key can happen here
4. Follow sequence to Section 27.7.9.1: Enable Encryption.

End State: _ENABLE = _YES; CRYPT_STS = _ACTIVE

27.7.9.3 Programming Interface

Priv interface

To protect HDCP 2.2 registers from tampering, accesses to them are limited to the SE secure processors.

Control Register

The SOR_NV_PDISP_SOR_HDCP22_CTRL register controls the Encryption of the actual data and initializing the HDCP variables. Refer to Section 27.8.224: SOR_NV_PDISP_SOR_HDCP22_CTRL_0for a complete description of this register.

AES-CTR Key Bus

The SOR_NV_PDISP_SOR_HDCP22_AES_CTR_KEY_[MSB/LSB1/LSB2/LSB3]_0 registers hold the Key value to drive the AES-CTR. These registers are programmed by software to the Session key value. This value will be XORed with the Ic128. These registers must be set to correct Ks value before setting the NV_PDISP_SOR_HDCP22_CTRL_ENABLE to _YES.

Refer to Section 27.8.226: SOR_NV_PDISP_SOR_HDCP22_AES_CTR_KEY_MSB_0 through Section 27.8.229: SOR_NV_PDISP_SOR_HDCP22_AES_CTR_KEY_LSB3_0 for complete descriptions of these registers.

AES-CTR Data Bus

The SOR_NV_PDISP_SOR_HDCP22_AES_CTR_DATA_MSB_0 and SOR_NV_PDISP_SOR_HDCP22_AES_CTR_-DATA_LSB_0 registers hold the Data input value to drive the AES-CTR. These registers are programmed by software to 64 bits riv value. For HDMI, this value is concatenated by inputCtr value for specific protocol. For DP, this value is XORed with TYPE value. This register must be set to the correct riv value before setting NV_PDISP_SOR_HDCP22_CTRL_ENABLE to _YES.

Refer to Section 27.8.230: SOR_NV_PDISP_SOR_HDCP22_AES_CTR_DATA_MSB_0 through Section 27.8.231: SOR_NV_PDISP_SOR_HDCP22_AES_CTR_DATA_LSB_0 for complete descriptions of these registers.

AES-CTR DP Type

The SOR_NV_PDISP_SOR_HDCP22_SST_DP_TYPE_0 register is used when the SOR is set to DP protocol. It allows software to pass the Stream Type value to the hardware which is used during encryption.

Refer to Section 27.8.232: SOR_NV_PDISP_SOR_HDCP22_SST_DP_TYPE_0 for a complete description of this register.

27.7.10 Tegra X1 Display Timing Constraints

Programming of display timing registers must meet these restrictions:

- Constraint 1: H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH > 20.
- Constraint 2: V_REF_TO_SYNC + V_SYNC_WIDTH + V_BACK_PORCH > 1.
- Constraint 3: V_FRONT_PORCH + V_SYNC_WIDTH + V_BACK_PORCH > 1 (vertical blank).

- Constraint 4: V_SYNC_WIDTH >= 1

  H_SYNC_WIDTH >= 1
- Constraint 5: V_REF_TO_SYNC >= 1

  H_REF_TO_SYNC >= 0
- Constraint 6: V_FRONT_PORCH >= (V_REF_TO_SYNC + 1)

  H_FRONT_PORCH >= (H_REF_TO_SYNC + 1)
- Constraint 7: H_DISP_ACTIVE >= 16

  V_DISP_ACTIVE >= 16

27.8 Display Interfaces Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

27.8.1 SOR_CTXSW_0

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this: Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts. Another way to avoid context switch interrupts is to set the AUTO_ACK bit.

This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

27.8.2 SOR_NV_PDISP_SOR_SUPER_STATE0_0

The following 2 registers are the equivalent for supervisor methods that dGPU DSI hardware generates. In Tegra X1, software must generate them.

Writing a 1 to this field cause a 1 cycle pulse which is used to activate the fields in registers NV_PDISP_SOR_STATE[1].

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UPDATE

27.8.3 SOR_NV_PDISP_SOR_SUPER_STATE1_0

NV_PDISP_SOR_SUPER_STATE1 contains triple buffered registers. The values written to SUPER_STATE1 are assembly values.

These are promoted to ARMed state when SUPER_STATE0 UPDATE is written.

The ARMed values are then promoted to ACTIVE internally by HW when an internally generated LOADV signal arrives.

The active state is reported in DISP_SOR_PWR and DISP_SOR_TEST registers

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	NO	ATTACHED: YES: Attach SOR to a display head NO: Detach SOR from display head 0 = NO 1 = YES
2	SAFE	ASY_ORMODE: SAFE: SOR is in safe low power state and not sending any active data. NORMAL: SOR is actively sending data 0 = SAFE 1 = NORMAL
1:0	0x0	ASY_HEAD_OPMODE: SLEEP: Display is not sending pixels and SOR will stop reading pixels from FIFOs SNOOZE: This should never be set by SW AWAKE: Display is sending active pixels to SOR 0 = SLEEP 1 = SNOOZE 2 = AWAKE

27.8.4 SOR_NV_PDISP_SOR_STATE0_0

Writing a 1 to this field cause a 1 cycle pulse which is used to promote the activate the fields in registers NV_PDISP_SOR_STATE[1], NV_PDISP_HEAD_STATE[0-5].

Offset: 0x3 | Byte Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UPDATE

27.8.5 SOR_NV_PDISP_SOR_STATE1_0

SOR Control Register

Offset: 0x4 | Byte Offset: 0x10 | Read/Write: R/W | Reset: 0x00000040 (0bxxxxxxxxxxxx00000000000000001000000)

Bit	Reset	Description
20:17	DEFAULTVAL	ASY_PIXELDEPTH: This sets the pixel depth and the only valid values are DEFAULTVAL for HDMI mode and BPP_18_444 and BPP_24_444 for DP mode. 0 = DEFAULTVAL 2 = BPP_18_444 5 = BPP_24_444
16:15	OFF	ASY_REPLICATE: This used to enable/disable pixel replication for HDMI. 0 = OFF 1 = X2 2 = X4

Bit	Reset	Description
14	0x0	ASY_DEPOL: This field needs to be set based on the DE polarity as needed by the panel. 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
13	0x0	ASY_VSYNCPOL: This field needs to be set based on the VSYNC polarity as needed by the panel. 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
12	0x0	ASY_HSYNCPOL: This field needs to be set based on the HSYNC polarity as needed by the panel. 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
11:8	LVDS_CUSTOM	ASY_PROTOCOL: 0 = LVDS_CUSTOM 1 = SINGLE_TMDS_A 2 = SINGLE_TMDS_B 8 = DP_A 9 = DP_B 15 = CUSTOM
7:6	COMPLETE_RASTER	ASY_CRCMODE: This field controls the symbols (eDP) that get CRCed. ACTIVE_RASTER --> Only active regions of the raster are CRCed COMPLETE_RASTER --> Entire raster (active + blanking) will be CRCed. NON_ACTIVE_RASTER --> Only non-active regions of the raster are CRCed 0 = ACTIVE_RASTER 1 = COMPLETE_RASTER 2 = NON_ACTIVE_RASTER
5:4	NONE	ASY_SUBOWNER: This field should always set to NONE. 0 = NONE 1 = SUBHEAD0 2 = SUBHEAD1 3 = BOTH
3:0	NONE	ASY_OWNER: This field controls the display pipe that is being connected to the SOR. Note that the ARMED value of this field is promoted only when SOR gets attached. HEAD0 --> When SOR needs to be connected to display internal head: displaya HEAD1 --> When SOR needs to be connected to display external head: displayb NONE --> When it is not connected to any heads 0 = NONE 1 = HEAD0 2 = HEAD1

27.8.6 SOR_NV_PDISP_HEAD_STATE0_0

Head Control Register

Note: *HEAD_STATE registers are valid in DP mode only.*

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x5..0x6 | Byte Offset: 0x14..0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:4	PROGRESSIVE	INTERLACED: Should always be set to PROGRESSIVE because INTERLACED is not supported. 0 = PROGRESSIVE 1 = INTERLACED
3	DISABLE	RANGECOMPRESS: Compresses the range of an RGB signal with black at 0, white at nominal 255 to the range required for CEA ranged RGB output, i.e., black at 16, white at 235. This control gives the driver a very quick way to compress RGB levels to be compatible with those required for output to TV via the TV encoder and/or to HDMI. 0 = DISABLE 1 = ENABLE
2	VESA	DYNRANGE: Sets the dynamic range for the output. VESA is the full 0 to maximum range. CEA mode will clip the output to fit within CEA range. Display Port has a restriction that BPP_18_444 RGB mode must use VESA. 0 = VESA 1 = CEA

Bit	Reset	Description
1:0	RGB	COLORSPACE: Should always be set to RGB. YUV_601/YUV_709 are not supported. 0 = RGB 1 = YUV_601 2 = YUV_709

27.8.7 SOR_NV_PDISP_HEAD_STATE1_0

This register sets the size of the raster

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x7..0x8 | Byte Offset: 0x1c..0x20 | Read/Write: R/W | Reset: 0x01011000 (0bx000000100000001x001000000000000)

Bit	Reset	Description
30:16	0x101	VTOTAL: This field is the total number of lines in a frame of the raster.
14:0	0x1000	HTOTAL: This field is the total number of pixels in a line of the raster.

27.8.8 SOR_NV_PDISP_HEAD_STATE2_0

This register sets the location of the horizontal and vertical sync end

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0x9..0xa | Byte Offset: 0x24..0x28 | Read/Write: R/W | Reset: 0x00000001 (0bx000000000000000x000000000000001)

Bit	Reset	Description
30:16	0x0	VSYNC_END: This field is the total number of lines after which vertical sync ends
14:0	0x1	HSYNC_END: This field is the total number of pixels after which horizontal sync ends

27.8.9 SOR_NV_PDISP_HEAD_STATE3_0

This register sets the location of horizontal and vertical blank end.

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0xb..0xc | Byte Offset: 0x2c..0x30 | Read/Write: R/W | Reset: 0x00010011 (0bx000000000000001x000000000010001)

Bit	Reset	Description
30:16	0x1	VBLANK_END: This field is the total number of lines after which vertical blank ends
14:0	0x11	HBLANK_END: This field is the total number of pixels after which horizontal blank ends.

27.8.10 SOR_NV_PDISP_HEAD_STATE4_0

This register sets the location of the horizontal and vertical blank start.

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0xd..0xe | Byte Offset: 0x34..0x38 | Read/Write: R/W | Reset: 0x00110100 (0bx000000000010001x000000100000000)

Bit	Reset	Description
30:16	0x11	VBLANK_START
14:0	0x100	HBLANK_START

27.8.11 SOR_NV_PDISP_HEAD_STATE5_0

The following fields need to be set only for interlaced mode

This is an array of 2 identical register entries; the register fields below apply to each entry.

Offset: 0xf..0x10 | Byte Offset: 0x3c..0x40 | Read/Write: R/W | Reset: 0x00000001 (0bx000000000000000x000000000000001)

Bit	Reset	Description
30:16	0x0	VBLANK_END_2
14:0	0x1	VBLANK_START_2

27.8.12 SOR_NV_PDISP_SOR_CRC_CNTRL_0

CRC_CONTROL

The main CRC enable bit flows along the pipeline. This register controls the injection of this bit at the top of the pipeline.

This register is double-buffered. The active value is updated at start of each frame from this 'arm' register.

Other registers such as SOR_STATE2.ASY_CRCMODE and SOR_*CRC* control the actual CRC logic, once enabled.

ARM_CRC_CONTROL enables or disables computation of CRC, effective at the start of next frame.

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	NO	ARM_CRC_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

27.8.13 SOR_NV_PDISP_SOR_CLK_CNTRL_0

LINK_SPEED

Programs the link speed for DP, i.e. it selects the multiplier used by the analog macro PLL.

For DP and TMDS, logic in the SOR will receive a version of this clock that is divided by 10. This field should only be changed when MODE_BYPASS is set to DP_SAFE or when the SOR is in safe mode.

It may take up to 200 microseconds for the PLLs in the analog macro to settle after this setting is changed.

Changing this field will require DP link re-training.

G1_62: Selects the 1.62 GHz link clock; PLL generates 1.62 GHz from 270MHz (multiplier = 6)

G2_7: Selects the 2.70 GHz link clock; PLL generates 2.70 GHz from 270MHz (multiplier = 10)

G5_4: Selects the 5.40 GHz link clock; PLL generates 5.40 GHz from 270MHz (multiplier = 20)

_LVDS: Link speed is 7*pixel clock used for LVDS (multiplier = 7)

CLK_SEL

The DP_SINGLE_LVDS macro has four input clocks. This field selects which clock is used for the internal logic. When the SOR is operating in DP mode, this should be set to one of the _DPCLK settings. When the SOR is operating in LVDS mode, this should be set to one of the _PCLK settings. This field should only be changed when the SOR is asleep or SOR_CLK_SEL is SAFE_SORCLK.

Please allow 200 microseconds for the PLLs in the macro to settle after changing this setting.

DIFF_PCLK and DIFF_PCLK inputs to macro are tied off. So software should only set to _SINGLE_(DP|P)CLK

_SINGLE_PCLK: Single ended pclk from the VPLLs

_DIFF_PCLK: Differential Pclk

_SINGLE_DPCLK: Single ended 270 MHz clock

_DIFF_DPCLK: Differential 270 MHz clock

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0x0000001b (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0011011)

Bit	Reset	Description
6:2	G1_62	DP_LINK_SPEED: 6 = G1_62 10 = G2_7 7 = LVDS 20 = G5_4 29 = G8_1 8 = G2_16 9 = G2_43 12 = G3_24 16 = G4_32
1:0	DIFF_DPCLK	DP_CLK_SEL: 0 = SINGLE_PCLK 1 = DIFF_PCLK 2 = SINGLE_DPCLK 3 = DIFF_DPCLK

27.8.14 SOR_NV_PDISP_SOR_CAP_0

Serial output resource

The registers are defined as if all the SORs are fully featured (e.g., dual link). If a feature is not supported, the register remains accessible i.e., writes will not hang, reads will always return the default value.

Each SOR unit consists of two sub-links designated A and B. In single link operation, the A and/or B sub-link is active. In dual link modes, both links are active. SOR units consist of either one or two links operating in the following combinations:

Sub-link A (single link mode)

Sub-link B (single link mode)

Sub-link AB (dual single link copy mode)

Sub-links A+B (dual link mode)

Sub-links A,B (dual single link mode) planned for future enhanced mode.

Driver level control of the SOR is accomplished through methods in the core channel of the display engine. However, certain aspects of SOR operation are chip, process, voltage, and pixel clock dependent. These need to be controlled via registers by the VBIOS or RM during mode configuration.

This register reports the innate capabilities of the SOR. Note that depending on the actual board build configuration, these values may not be exactly the same ones reflected in the overall capabilities structure reported through the class. With the exception of the last bit, the values assumed by these fields are automatically produced by the hardware.

Note: *THIS REGISTER DEFINITION MUST BE KEPT COHERENT WITH DSI_SOR_CAP(i) !*

Usage: boot / initialization

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	LVDS_ONLY: 0 = FALSE 1 = TRUE
25	X	DP_B: 0 = FALSE 1 = TRUE
24	X	DP_A: 0 = FALSE 1 = TRUE

Bit	Reset	Description
20	X	DDI: 0 = FALSE 1 = TRUE
16	X	SDI: 0 = FALSE 1 = TRUE
13	X	DISPLAY_OVER_PCIE: 0 = FALSE 1 = TRUE
12	X	SINGLE_TMDS_225_MHZ: 0 = FALSE 1 = TRUE
11	X	DUAL_TMDS: 0 = FALSE 1 = TRUE
10	X	DUAL_SINGLE_TMDS: 0 = FALSE 1 = TRUE
9	X	SINGLE_TMDS_B: 0 = FALSE 1 = TRUE
8	X	SINGLE_TMDS_A: 0 = FALSE 1 = TRUE
3	X	DUAL_LVDS_24: 0 = FALSE 1 = TRUE
2	X	DUAL_LVDS_18: 0 = FALSE 1 = TRUE
1	X	SINGLE_LVDS_24: 0 = FALSE 1 = TRUE
0	X	SINGLE_LVDS_18: 0 = FALSE 1 = TRUE

27.8.15 SOR_NV_PDISP_SOR_PWR_0

This register contains bits that control the power state of the SOR. For simplicity, the sequencer will be used to perform all power control operations in the SOR (even in TMDS where the sequencing is relatively simple). This unifies the approach and makes it easier for the hardware and software to deal with the SOR's. At boot, the software must load and configure the SOR sequencer via the SOR_SEQ_CTL and SOR_SEQ_INST registers below. The sequencer must be properly programmed for power control to work.

NORMAL_STATE: Sets the normal operating state. There are two choices: powered up and powered down. To change this register, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, SETTING_NEW will indicate DONE. This is the state that software will want to control in order to power up and down the interface.

NORMAL_START: The execution of the power up and power down sequence can be modified somewhat by choosing to use either the NORM or ALT program entry point.

SAFE_STATE: Sets the safe operating state. There are only two choices: powered up and powered down. To change this register, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, the SETTING_NEW will indicate DONE. The execution of the power up and power down sequence can be modified somewhat by choosing to use either the standard or the alternate program entry point. This is the state that the hardware will use whenever it initiates the mode switch/shutdown procedure for HDMI. In general, this should be set to STATE_PD and left there always.

SAFE_START: The execution of the power up and power down sequence can be modified somewhat by choosing to use either the NORM or ALT program entry point.

HALT_DELAY: Once the sequencer gets to the instruction with the HALT=1, the program is essentially complete; that is, the attached unit is powered up or down as was requested. Some panels, however, have a minimum time before their state can be changed again. If the halt instruction has a non-zero delay, this bit will be set while that time expires.

MODE: The currently active state, normal or safe. After reset, the MODE is always SAFE.

SETTING_NEW: This bit is used to trigger a new setting of power mode to take effect. The typical procedure might be something like:

1. Make sure SETTING_NEW==DONE, i.e., not already an outstanding change request. If there is an outstanding change request, software must wait for it to complete.
2. Update the NORMAL and SAFE power mode fields.
3. Write SETTING_NEW=TRIGGER.
4. Poll for SETTING_NEW=DONE. If it does not happen within one frame time, then software may opt to accelerate the change by writing SOR_SEQ_CTL SWITCH=FORCE (be careful not to disturb other settings in that register!).

Note: *Software can start polling on SEQ_CTL_STATUS right after writing SETTING_NEW to trigger because SEQ_CTL_STATUS is set to RUNNING after 2 sorclk cycles which will always be less than the register read delay.*

Usage: boot / initialization / mode switch / normal operation / shutdown

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: R/W | Reset: 0xXX000000 (0b0xxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
31	RW	DONE	SETTING_NEW:w1c 0 = DONE 1 = PENDING 1 = TRIGGER
28	RO	X	MODE: 0 = NORMAL 1 = SAFE
24	RO	X	HALT_DELAY: 0 = DONE 1 = ACTIVE
17	RW	NORMAL	SAFE_START: 0 = NORMAL 1 = ALT
16	RW	PD	SAFE_STATE: 0 = PD 1 = PU
1	RW	NORMAL	NORMAL_START: 0 = NORMAL 1 = ALT
0	RW	PD	NORMAL_STATE: 0 = PD 1 = PU

27.8.16 SOR_NV_PDISP_SOR_PLL0_0

These registers configure the main SOR PLL and other frequency-dependent controls. The value loaded is a function of the pixel clock frequency, and whenever the pixel clock changes, these registers must be updated. In general, these registers will be updated during the second interrupt of a mode switch.

RESISTORSEL: Selects the internal resistor (0) or external resistor (1).

PULLDOWN: Weak pull-down enable

- 1 = 2Kohm pulldown on all outputs.
- 0 = Weak pulldown disabled.

Note: *Pulldown is also controlled by the sequencer. the pulldown is derived from an OR of:*

NV_PDISP_SOR_PLL0_PULLDOWN and the sequencer control of pulldown. This priv register field does not read the final pull-down value. In other words if software writes NV_PDISP_SOR_PLL0_PULLDOWN to DISABLED and sequencer pulldown is enabled, then an NV_PDISP_SOR_PLL0_PULLDOWN read will return DISABLED (while pulldown might be ENABLED due to sequencer control).

VCOCAP[3:0]: Selects the VCO capacitor and adjusts ring oscillator inter-stage load.

FILTER[3:0]

- Bits 2:0 select the loop filter and adjust the filter resistor value.
- Bit 3 controls this VCO startup bit for the TMDS_DUAL macro.
  - 0 - normal operation (default)
  - 1 - forced VCO oscillation mode

ICHPMP[3:0]: Specifies additions to the charge pump current in steps of 0.375 µA.

IOCURRENT[5:0]: Used for I/O control. This field has been replaced by the SOR_LANE_DRIVE_CURRENT registers.

TMDS_TERM: This bit is used to enable termination. Termination is only used in TMDS mode of operation, and may not be needed at lower operating frequencies (in which case, disabling it saves power).

TERMADJ[3:0]: Termination resistance control.

- 0000 - lowest

  ...
- 1000 - default

  ...
- 1111 - highest

COMPOUT: Termination calibration status. Only exists on DisplayPort SORs. This procedure should be performed on boot time to program TERMADJ correctly. Ideally this should be re-calibrated on Hot Plug Detect as well.

Procedure for termination calibration:

1. Start with the TERM_ADJ = 1000
2. Wait 100 µs and check COMPOUT
3. If COMPOUT=1, change the MSB to 0
4. Repeat for the next significant bits until all 4 bits are determined.

SUPPLYLEV[1:0]: Supply voltage level descriptor

LOADADJ[3:0]: Load pulse position adjust

LVDS_CM[1:0]: Reserved

COHERENTMODE: Output reference clock selector

0 = Non-coherent mode

1 = Coherent mode

BGAP_CNTL

Band-gap current setting control.

00 = Normal

01 = Positive coefficient

10 = Negative coefficient

11 = Normal + 12.5%

LOCKDET: Status signal indicating whether PLL is locked within the desired resolution.

0 = Not locked

1 = Locked

MISC_CNTL: Reserved.

DCIR_PLL_RESET: For the Display-over-PCIe® feature, a different type of analog macro was used in the SOR. The PLL in this macro needs to be held in reset during mode switches when the reference clock drops to safe-clock. A signal from the DCIR asserts this reset. NV_PDISP_SOR_PLL2_DCIR_PLL_RESET can be used to override this signal

OVERRIDE: The signal from the DCIR cannot power down the PLL.

ALLOW: The signal from the DCIR will be able to power down the PLL.

AUX0-AUX7: Most of these bits currently have no assigned function, but are provided to permit control of possible future features of the macro.

- AUX0: gate seq_2all_pll_pulldown from PULLDOWN. this can be used to remove sequencer control to PULLDOWN.
  - SEQ_PLL_PULLDOWN_OVERRIDE: Mask the pll_pulldown signal from the SOR sequencer so that it has no effect on the PLL_PULLDOWN port.
  - SEQ_PLL_PULLDOWN_ALLOW: Allow the pll_pulldown signal from the SOR sequencer to affect the PLL_PULLDOWN port.
- AUX1: Power on override for PLLCAPPD. The PLL in the analog macro will not run unless this is powered on. This can be used if the PLL needs to be powered on before the sequencer runs.
  - SEQ_PLLCAPPD_OVERRIDE: Mask the pll_reset signal from the SOR sequencer so that it has no effect on the PLLCPAPPD port of the SOR macro.
  - SEQ_PLLCAPPD_ALLOW: Allow the pll_reset signal from the SOR sequencer to assert PLLCAPPD.
- AUX2: gate PDBG from sequencer_pd_macro.

This can be used to force the Bandgap to power on.

  - OVERRIDE_POWERDOWN: gate the PDBG signal from the sequencer, PDBG can still be set by AUX6
  - ALLOW_POWERDOWN: normal operation, the PDBG signal from the sequencer can power down the bandgap
- AUX3: rotate green channel by 1 bit to reduce TMDS EMI
  - ROTATE_DISABLE: Do not rotate
  - ROTATE_ENABLE: Right-rotate green channel by 1 bit
- AUX4 has been used to enable duplicate controls for the Dual link HDCP. This is needed for some Dual Link panels.
  - DUPLICATE_CTRL_ENABLE: Copy the control bits from the primary link to the secondary.
  - DUPLICATE_CTRL_DISABLE: Zero out the ctrl bits on the secondary link.
- AUX5: No function
- AUX6: Main power down for bandgap and reference current setting circuitry
  (NOTE: This must be BANDGAP_POWERDOWN_DISABLE when in power down mode if 3.3V is seen on the IO_[A/B]_VDD rails)
  - BANDGAP_POWERDOWN_DISABLE: Allow the bandgap to powerup.
  - BANDGAP_POWERDOWN_ENABLE: Force the bandgap to powerdown.

- AUX7: Controls PDPORT in the analog macro which powers down all the output links/lanes; for instance, txd{n/p}0-9 for dual TMDS macro.
  - PORT_POWERDOWN_DISABLE: Allow the output ports to be turned on.
  - PORT_POWERDOWN_ENABLE: Force the output ports to power down. This will prevent any data from reaching the sink device.
- AUX8: No function.
- AUX9: No function.

Usage: boot / initialization / mode switch

Offset: 0x17 | Byte Offset: 0x5c | Read/Write: R/W | Reset: 0x050003c5 (0bxxxx0101xxxx0000xx000011110xx1x1)

Bit	Reset	Description
27:24	RST	ICHPMP: 5 = RST
19:16	RST	FILTER: 0 = RST
13:12	V25	TXREG_LEVEL: 0 = V25 1 = V15 2 = V35 3 = V45
11:8	RST	VCOCAP: 3 = RST
7:6	V45	PLLREG_LEVEL: 0 = V25 1 = V15 2 = V35 3 = V45
5	DISABLE	PULLDOWN: Set to DISABLE. 0 = DISABLE 1 = ENABLE
2	ASSERT	VCOPD: 0 = RESCIND 1 = ASSERT
0	OFF	PWR: 0 = ON 1 = OFF

27.8.17 SOR_NV_PDISP_SOR_PLL1_0

Usage: boot / initialization / mode switch

Offset: 0x18 | Byte Offset: 0x60 | Read/Write: R/W | Reset: 0x0000X000 (0bxx0xxx000000xxxxxxx10000xx000000)

Bit	R/W	Reset	Description
29	RW	DISABLE	COHERENTMODE: 0 = DISABLE 1 = ENABLE
25:24	RW	0x0	LVDSCM: Reserved
23:20	RW	CENTER	LOADADJ: 0 = CENTER
15	RO	X	TERM_COMPOUT: 0 = LOW 1 = HIGH
12:9	RW	OHM500	TMDS_TERMADJ: 8 = OHM500
8	RW	DISABLE	TMDS_TERM: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
5:0	RW	RST	IOCURRENT: 0 = RST

27.8.18 SOR_NV_PDISP_SOR_PLL2_0

Usage: boot / initialization / mode switch

Offset: 0x19 | Byte Offset: 0x64 | Read/Write: R/W | Reset: 0x05c00014 (0b00000101110000000000000000010100)

Bit	Reset	Description
31:28	0x0	PLL_MDIV
27:26	DP_TMDS	CLKGEN_MODE: 0 = Reserved 1 = DP_TMDS 2 = HBR3 3 = MPHY
25	LVDSEN_ALLOW	AUX9: 0 = LVDSEN_ALLOW 1 = LVDSEN_OVERRIDE
24	SEQ_PLLCAPPD_ENFORCE_ENABLE	AUX8: 0 = SEQ_PLLCAPPD_ENFORCE_DISABLE 1 = SEQ_PLLCAPPD_ENFORCE_ENABLE
23	PORT_POWERDOWN_ENABLE	AUX7: 0 = PORT_POWERDOWN_DISABLE 1 = PORT_POWERDOWN_ENABLE
22	BANDGAP_POWERDOWN_ENABLE	AUX6: 0 = BANDGAP_POWERDOWN_DISABLE 1 = BANDGAP_POWERDOWN_ENABLE
21	SINGLE_LINK_LVDS	AUX5: 0 = SINGLE_LINK_LVDS 1 = DUAL_LINK_LVDS
20	DUPLICATE_CTRL_DISABLE	AUX4: 0 = DUPLICATE_CTRL_DISABLE 1 = DUPLICATE_CTRL_ENABLE
19	ROTATE_DISABLE	AUX3: 0 = ROTATE_DISABLE 1 = ROTATE_ENABLE
18	OVERRIDE_POWERDOWN	AUX2: 0 = OVERRIDE_POWERDOWN 1 = ALLOW_POWERDOWN
17	SEQ_PLLCAPPD_ALLOW	AUX1: 0 = SEQ_PLLCAPPD_ALLOW 1 = SEQ_PLLCAPPD_OVERRIDE
16	SEQ_PLL_PULLDOWN_ALLOW	AUX0: 0 = SEQ_PLL_PULLDOWN_ALLOW 1 = SEQ_PLL_PULLDOWN_OVERRIDE
15:8	BY_1	PLL_NDIV: 0 = BY_1 1 = BY_1_1 2 = BY_2 3 = BY_3 4 = BY_4 5 = BY_5 6 = BY_6 7 = BY_7 8 = BY_8 9 = BY_9 10 = BY_10 11 = BY_11 12 = BY_12 13 = BY_13 14 = BY_14 15 = BY_15

Bit	Reset	Description
7:4	BY_2	PLL_PDIV: 0 = BY_1 1 = BY_2 2 = BY_4 3 = BY_8 4 = BY_16
3:2	TMDS_DP_MODE	PLL_PDIV_MODE: 0 = LVDS_MODE 1 = TMDS_DP_MODE 2 = EDP_RATE3_MODE
1	DISABLE	DIV_RATIO_OVERRIDE: 0 = DISABLE 1 = ENABLE
0	OVERRIDE	DCIR_PLL_RESET: 0 = OVERRIDE 1 = ALLOW

27.8.19 SOR_NV_PDISP_SOR_PLL3_0

SOR_PLL3

This register directly controls the SOR analog macro. This register will either be programmed to fixed value on boot, or will need to be set in the SOR IED scripts.

KICKSTART

_DISABLE: Disable kickstart

_LOOP_40: short loop-filter 40% of avdd14

_LOOP_60: short loop-filter 60% of avdd14

_LOOP_50: short loop-filter 50% of avdd14

AVDD14_LEVEL

Internal regulated 1.4v voltage level control bits

_1_35V: 1.35V

_1_40V: 1.40V

_1_45V: 1.45V

_1_50V: 1.50V

No other values are valid for this field.

AVDD10_LEVEL

Internal regulated 1.0v voltage level control bits

_0_95V: 0.95V

_1_00V: 1.00V

_1_05V: 1.05V

_1_10V: 1.10V

No other values are valid for this field.

CLKDIST_MODE

Determining the clock distribution by CMOS buffer or CML buffers

_CMOS: CMOS buffers

_CML: CML buffers

PLLVDD_MODE

Field to determine the PLL voltage. Software needs to program this after power-on.

_3_3V: 3.3V --> For eDP

PLL_BYPASS

Set to DISABLE.

_ENABLE: Force the PLL output onto all lanes.

_DISABLE: Normal operation

TEST_REFCLK_EN

Software should not use this field.

BG_VREF_LEVEL

Control bits for the bandgap's output voltages

V(bandgap output) = 0.7v * (84% + BG_VREF_LEVEL*2%)

i.e.

0000 -> 0.588v

...

1000 -> 0.7v

...

1111 -> 0.798v

BG_TEMP_COEF

Control bits for the bandgap's temperature coefficient

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x38000440 (0b00111000000000000x00001000100xx00)

Bit	Reset	Description
31:28	0x3	BG_TEMP_COEF
27:24	0x8	BG_VREF_LEVEL
23:16	0x0	TEST_REFCLK_EN
14	DISABLE	PLL_BYPASS: 0 = DISABLE 1 = ENABLE
13	V1_8	PLLVDD_MODE: 0 = V1_8 1 = V3_3
12	CMOS	CLKDIST_MODE: 0 = CMOS 1 = CML
11:8	V1_00	AVDD10_LEVEL: 0 = V0_80 1 = V0_85 2 = V0_90 3 = V0_95 4 = V1_00 5 = V1_05 6 = V1_10 7 = V1_15

Bit	Reset	Description
7:4	V1_40	AVDD14_LEVEL: 0 = V1_12 1 = V1_19 2 = V1_26 3 = V1_33 4 = V1_40 5 = V1_47 6 = V1_54 7 = V1_61
1:0	DISABLE	KICKSTART: 0 = DISABLE 1 = LOOP_40 2 = LOOP_50 3 = LOOP_60

27.8.20 SOR_NV_PDISP_SOR_CSTM_0

The class permits the driver to select a number of operating modes for the SOR. Some of these modes (TMDS modes) are well defined and stable. Some modes may require customization by software before they will work. These registers are used for that customization. The fields available for customization are:

- PD_TXDA[3:0]: Bitwise control to power down the data pins of link A. Set to DISABLE to power down the pin.
- PD_TXDB[3:0]: Bitwise control to power down the data pins of link B. Set to DISABLE to power down the pin.
- PD_TXCA: Power down the clock pin of link A. Set to DISABLE to power down the pin.
- PD_TXCB: Power down the clock pin of link B. Set to DISABLE to power down the pin.

UPPER: Designates whether LVDS bank A is the upper, odd, or first pixel. Bank B is always set to !UPPER. The serial output from UPPER=1 will be Clock and A0-A3. The serial output from UPPER=0 (and DUALMODE) will be Clock and A4-A7. The default is bank A has UPPER=1, bank B has UPPER=0.

MODE[1:0]: Controls the digital output encoding applied to the data stream in custom mode and is only used for data muxing at the input of the SOR. This field does not control the TMDS macro.

- 0 = Reserved
- 1 = TMDS
- 2 = Reserved
- 3 = Reserved

LINKACTA, LINKACTB: Enables (1) or disables (0) the digital logic of links A and B

LVDS_EN: Output driver configuration for controlling the encoding of the data and the output common mode control. This does not control the internal clock dividers of the TMDS macro.

- 0 = TMDS
- 1 = Reserved

DUP_SYNC: Reserved

NEW_MODE: For backwards compatibility, set register to 0 so the old mode is used. In old mode, for the second link in dual link mode, all the control bits are zeroed out except for VSYNC. When a new mode is used, none of the control bits of the second link for dual-link mode are zeroed out.

BALANCED: Default is unbalanced. Has no effect in other modes.

PLLDIV: Controls the internal clock dividers of the TMDS_MACRO by setting the feedback divider for the high-speed PLL.

- 0 = Reserved
- 1 = divide by 10 (TMDS)

ROTCLK[3:0]: Skews the TXC clock to come out earlier. By changing this register value, you can configure the skew between output data (TX data) and output clock (TXC). This field specifies the number of sclk cycles which the output clock should come out earlier than its normal phase. For TMDS, sclk is a 10x pixel clock, so ROTVAL should be between 0-9. For LVDS, sclk is a 7x pixel clock, so ROTVAL should be between 0-7.

ROTDAT[2:0]: Before encoding the 8 bits of each color channel, the 8 bits within each color channel can be right rotated. All color channels are rotated by the same amount. For example, if ROTDAT = 6, then input channel data (r7,r6,r5,r4,r3,r2,r1,r0} would become {r5,r4,r3,r2,r1,r1,r7,r6}.

ROTDAT should be between 0 and 7.

TMDS modes of operation are standard and stable. The table below summarizes the fixed values the hardware uses for the above fields for TMDS operating modes.

	SINGLE TMDS_A	SINGLE TMDS_B	SINGLE TMDS_AB	SINGLE TMDS	Dual TMDS
PD_TXDA[2:0]	0	7	0	0	0
PD_TXDA[3]	1	1	1	1	1
PD_TXDB[2:0]	7	0	0	0	0
PD_TXDB[3]	1	1	1	1	1
PD_TXCA	0	1	0	0	0
PD_TXCB	1	0	0	0	1
UPPER	1	1	1	1	1
MODE[1:0]	1	1	1	1	1
LINKACTA	1	0	1	1	1
LINKACTB	0	1	1	1	1
LVDS_EN	0	0	0	0	0
DUP_SYNC	0	0	0	0	0
NEW_MODE	0?	1	1	1	0?
BALANCED	0	0	0	0	0
PLLDIV	1	1	1	1	1
ROTCLK[3:0]	0	0	0	0	0
ROTDAT[2:0]	0	0	0	0	0

Where X (ROTCLK[3:0]) = NV_PDISP_SOR_CSTM_ROTCLK;

The first register defines the total custom mode. This is useful for defining possible new modes of operation. Which is to say, this is a debug register set. Software should really never need to use it. Also, note that if LVDS_ONLY=TRUE, then this register cannot be used. This register is intended to have the same format as the second register.

Usage: boot / initialization / mode switch

Offset: 0x1b | Byte Offset: 0x6c | Read/Write: R/W | Reset: 0x0001c800 (0bx0000000xx0x000111001x0000000000)

Bit	Reset	Description
30:28	RST	ROTDAT: 0 = RST
27:24	RST	ROTCLK: 0 = RST
21	BY_7	PLLDIV: 0 = BY_7 1 = BY_10
19	DISABLE	BALANCED: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	DISABLE	NEW_MODE: 0 = DISABLE 1 = ENABLE
17	DISABLE	DUP_SYNC: 0 = DISABLE 1 = ENABLE
16	ENABLE	LVDS_EN: 0 = DISABLE 1 = ENABLE
15	ENABLE	LINKACTB: 0 = DISABLE 1 = ENABLE
14	ENABLE	LINKACTA: 0 = DISABLE 1 = ENABLE
13:12	LVDS	MODE: 0 = LVDS 1 = DP
11	TRUE	UPPER: 0 = FALSE 1 = TRUE
9	ENABLE	PD_TXCB: 0 = ENABLE 1 = DISABLE
8	ENABLE	PD_TXCA: 0 = ENABLE 1 = DISABLE
7	ENABLE	PD_TXDB_3: 0 = ENABLE 1 = DISABLE
6	ENABLE	PD_TXDB_2: 0 = ENABLE 1 = DISABLE
5	ENABLE	PD_TXDB_1: 0 = ENABLE 1 = DISABLE
4	ENABLE	PD_TXDB_0: 0 = ENABLE 1 = DISABLE
3	ENABLE	PD_TXDA_3: 0 = ENABLE 1 = DISABLE
2	ENABLE	PD_TXDA_2: 0 = ENABLE 1 = DISABLE
1	ENABLE	PD_TXDA_1: 0 = ENABLE 1 = DISABLE
0	ENABLE	PD_TXDA_0: 0 = ENABLE 1 = DISABLE

27.8.21 SOR_NV_PDISP_SOR_BLANK_0

This register can be used to override the SOR output resource pixels with blank data.

OVERRIDE: Setting this field to true will override the pixel bus from the RG and output black pixels instead.

TRANSITION

This field controls the timing of the output resource blank override. The choices are

- IMMEDIATE The output resource will be blanked or restored to its previous output data immediately.

- NEXT_VSYNC The output resource will be blanked or restored to its previous output data at the next vsync.

STATUS This read-only field returns BLANKED when the output resource is sending blank pixels forced by the OVERRIDE bit, otherwise it returns NOT_BLANKED.

Usage: boot / initialization / mode switch / normal operation

Offset: 0x1f | Byte Offset: 0x7c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
2	RO	X	STATUS: 0 = NOT_BLANKED 1 = BLANKED
1	RW	IMMEDIATE	TRANSITION: 0 = IMMEDIATE 1 = NEXT_VSYNC
0	RW	FALSE	OVERRIDE: 0 = FALSE 1 = TRUE

27.8.22 SOR_NV_PDISP_SOR_SEQ_CTL_0

Sequencer control registers for SOR. Up to three pins can be assigned to an SOR for use in controlling the power to an attached LVDS flat panel. The meaning of any particular pin and whether it is actually available to this SOR is controlled in either the host or PCB manager. In addition, the sequencer can override the individual link clock and data power control pins, thereby forcing them all into disabled or tristate mode and it can override the DE signal from the RG (for TMDS mode) to force the link to become inactive.

PU_PC: The program counter for the start of the power up program sequence. Always 0.

PU_PC_ALT: The alternate entry point into the power up program sequence. Defaults to the same value as PU_PC.

PD_PC: The program counter for the start of the power down program sequence. Defaults to 8, evenly dividing the available program space.

PD_PC_ALT: The alternate entry point into the power down program sequence. Defaults to the same default value as PD_PC.

PC: The current value of the program counter (useful for status and debug).

STATUS: Indicates if the sequencer is STOPPED or RUNNING.

SWITCH: If a particular sequencer instruction is waiting for VSYNC to arrive, writing this bit to FORCE will cause the wait condition to be satisfied immediately.

The sequencer can be run both when the SOR is attached and detached. However, when sor is detached WAIT_UNITS_VSYNC is not honored. Also, BLANK_DATA, BLANK_DE, BLANK_V and BLANK_H are ignored.

NV_PDISP_SOR_DP_LINKCTL_ENABLE no longer controls PDBG or PLLCAPPD. The SOR sequencer or NV_PDISP_SOR_PLL2 should be used for these control bits.

Usage: boot / initialization / mode switch

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0xX00X880X (0bxxxxxxxxxxxxxxxx100010000000xxxx)

Bit	R/W	Reset	Description
30	RW	X	SWITCH: 0 = WAIT 1 = FORCE
28	RO	X	STATUS: 0 = STOPPED 1 = RUNNING
19:16	RO	X	PC
15:12	RW	0x8	PD_PC_ALT
11:8	RW	0x8	PD_PC

Bit	R/W	Reset	Description
7:4	RW	0x0	PU_PC_ALT
3:0	RO	X	PU_PC

27.8.23 SOR_NV_PDISP_SOR_LANE_SEQ_CTL_0

The SOR_LANE_SEQ_CTL register is used to control the operation of the SOR lane sequencer. The lane sequencer is used to power up or power down the lanes in the analog macro one at a time to prevent spikes on the power rail.

LANE_STATE

This is the current status of each lane's power.

	DP	TMDS	LVDS
LANE0	DPA lane2	LinkA Lane0 LinkA Lane0	
LANE1	DPA lane1	LinkA Lane1	LinkA Lane1
LANE2	DPA lane0	LinkA Lane2	LinkA Lane2
LANE3	unused	unused	LinkA Lane3
LANE4	DPA lane3	LinkA clock	LinkA clock
LANE5	DPB lane2	LinkB Lane0	LinkB Lane0
LANE6	DPB lane1	LinkB Lane1	LinkB Lane1
LANE7	DPB lane0	LinkB lane2	LinkB Lane2
LANE8	unused	unused	LinkB Lane3
LANE9	DPB lane3	LinkB clock	LinkB clock

DELAY

Number of microseconds to delay between each lanes' power state change.

The recommended value is 1us.

NEW_POWER_STATE

This controls whether the lanes should be powered up or powered down when this sequencer is triggered by SETTING_NEW_TRIGGER.

PU: Power up the requested lanes

PD: Power down to the requested lanes

SEQUENCE

Controls the direction of the power up or power down sequence.

UP: Lanes are powered up in increasing order

DOWN: Lanes are powered up in decreasing order

STATE

Whenever this sequencer is running, this field will be set to _BUSY.

SETTING_NEW

This bit can be used to run this sequencer outside of the normal SOR sequencer operation. If the sequencer is already BUSY because of a request from the SOR sequencer, the lane sequencer will wait until that request is complete before servicing this request. It is recommended that this trigger should only be used when the SOR sequencer is not running in order to prevent this type of interaction.

The lane sequencer looks at the current protocol, NV_PDISP_SOR_DP_LINKCTL_ENABLE and whether the request is power-up or power-down to determine what the new lane state should be.

DP_LINKCTL.ENABLE	NEW_POWER_STATE	Protocol	New Lane State(1=POWER DOWN)
0	0	LVDS	LVDS.PD_TXD* | ~LVDS.LINKACT | SEQ_INST.TRISTATE_IOS
0	0	CSTM	CSTM.PD_TXD* | ~CSTM.LINKACT | SEQ_INST.TRISTATE_IOS
0	1	!DP	1
X	X	DP	~DP_PADCTL.PD_TXD*
1	X	X	~DP_PADCTL.PD_TXD*

Usage: boot / initialization / mode switch

Offset: 0x21 | Byte Offset: 0x84 | Read/Write: R/W | Reset: 0xX0011XXX (0b0xxxxxxxxxx0xxx10001xxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DONE	SETTING_NEW: 0 = DONE 1 = PENDING 1 = TRIGGER
28	RO	X	SEQ_STATE: 0 = IDLE 1 = BUSY
20	RW	UP	SEQUENCE: 0 = UP 1 = DOWN
16	RW	PD	NEW_POWER_STATE: 0 = PU 1 = PD
15:12	RW	0x1	DELAY
9	RO	X	LANE9_STATE: 0 = POWERUP 1 = POWERDOWN
8	RO	X	LANE8_STATE: 0 = POWERUP 1 = POWERDOWN
7	RO	X	LANE7_STATE: 0 = POWERUP 1 = POWERDOWN
6	RO	X	LANE6_STATE: 0 = POWERUP 1 = POWERDOWN
5	RO	X	LANE5_STATE: 0 = POWERUP 1 = POWERDOWN
4	RO	X	LANE4_STATE: 0 = POWERUP 1 = POWERDOWN
3	RO	X	LANE3_STATE: 0 = POWERUP 1 = POWERDOWN
2	RO	X	LANE2_STATE: 0 = POWERUP 1 = POWERDOWN
1	RO	X	LANE1_STATE: 0 = POWERUP 1 = POWERDOWN
0	RO	X	LANE0_STATE: 0 = POWERUP 1 = POWERDOWN

27.8.24 SOR_NV_PDISP_SOR_SEQ_INSTx_0

This register set is used to preload the power-up and power-down sequences. Each step of the program performs the following based on the fields below.

First, either delay for WAIT time or until the next leading edge of vertical sync from the RG. The range of WAIT is 0-1023 us or 0-1023 ms.

Second, set sequencer outputs (A, B, I/O pin powerdown override, etc.) to the values specified in the instruction. The meaning and use of pins A and B is chip and board dependent i.e., something the SW will need to know how to configure.

Lastly, if HALT==1, stop, otherwise, execute the next instruction.

WAIT_TIME: Amount of time to wait (in either us or ms). Note if WAIT_UNITS=VSYNC, the WAIT_TIME field is ignored.

WAIT_UNITS: Wait delay mode, us, ms, VSYNC.

PDPLL: Asserts the PDPLL port on the analog macro, powering down the pll. This sequencer control can be overridden by NV_PDISP_SOR_PLL2_AUX1_SEQ_PLLCAPPD_OVERRIDE. NV_PDISP_SOR_PLL0_PWR_OFF will force PDPLL to be asserted, regardless of the state of the sequencer.

YES: Power down the PLL.

NO: Do not power down the PLL

PDPORT: Asserts the PDPORT port on the SOR analog macro. This powers down all output lanes. This sequencer control can be overridden by NV_PDISP_SOR_DP_LINKCTL_ENABLE. NV_PDISP_SOR_PLL2_AUX7_PORT_POWERDOWN_ENABLE will force PDPORT to be asserted, regardless of the state of the sequencer.

YES: Power down the port

NO: Do not power down the port

LANE_SEQ: When set to _RUN, the sequencer will kick off the Lane Sequencer (see NV_PDISP_SOR_LANE_SEQ_CTL). The SOR sequencer will not continue to the next instruction until the lane sequencer completes. The lane sequencer will run before the other commands in the current instruction take effect and before the delay in the current instruction.

SEQUENCE: This field controls the direction of the lane power up or power down sequence.

UP: Lanes are powered up or down in increasing order

PDTXD_0, PDTXD_1 .... PDTXD_9

DOWN: Lanes are powered up or down in decreasing order

PDTXD_9, PDTXD_8 .... PDTXD_0

PIN_A, PIN_B: State to drive the external pin to. The connection state of these pins is controlled in either the host or the pcb manager.

DRIVE_PWM_OUT_LO: Drive the PWM circuit output lo. Note that this driven lo condition is applied before the optional PWM_POLARITY control, so that the external pin can actually be driven hi or lo.

TRISTATE_IOS: Coerce all output pins to tristate.

0 = enable output pins

1 = disable/tristate output pins

For chips with the SOR Lane Sequencer, this does not take effect until the Lane Sequencer is run. Since Lane Sequencer runs before the values in the current instruction take effect, TRISTATE_IOS should be set to the desired value on the instruction before LANE_SEQ_RUN.

BLACK_DATA: Override the pixel bus from the rg, and replace the pixels with black.

0 = normal data output

1 = force black output

BLANK_DE, BLANK_H, BLANK_V: Override the de, hsync, and vsync signals from the rg. The signals will be forced to the inactive i.e. deasserted state.

0 = normal output

1 = force signal inactive/deasserted

ASSERT_PLL_RESET: Assert reset to the PLL (PLLCAPPD) This sequencer control can be overridden by NV_PDISP_SOR_PLL2_AUX1_SEQ_PLLCAPPD_OVERRIDE.

NV_PDISP_SOR_PLL2_AUX8_SEQ_PLLCAPPD_ENFORCE_ENABLE will force PLLCAPPD to be asserted, regardless of the state of the sequencer.

0 = normal

1 = force reset

POWERDOWN_MACRO: Override LINKACTA, LINKACTB and force TMDS macro to powerdown state.

(Only controls PDBG)

0 = normal operation

1 = force powerdown

PULLDOWN: Weak pulldown enable (Only exists on TMDS SORs)

1 = 2Kohm pulldown on all outputs.

0 = Weak pulldown disabled.

Note: *These program examples are included here to illustrate the operation of the sequencer. They should not be assumed to be correct for actual operation.*

Power-up sequence LVDS (assume PIN_A controls panel power, PIN_B controls backlight).

Initial conditions (assume panel off = backlight off = 0):

HALT=0

PIN_A=0

PIN_B=0

DRIVE_PWM_OUT_LO=0

TRISTATE_IOS=1

BLACK_DATA=1

BLANK_DE=1

BLANK_H=1

BLANK_V=1

ASSERT_PLL_RESET=1

POWERDOWN_MACRO=1

Program

1. POWERDOWN_MACRO=0, WAIT ?
2. ASSERT_PLL_RESET=0, WAIT ?

3. PIN_A=1, WAIT ?
4. TRISTATE_IOS=0, WAIT ?
5. PIN_B=1, WAIT vsync
6. HALT=1, BLACK_DATA=0, BLANK_DE=0, BLANK_H=0, BLANK_V=0, WAIT ?

Powerdown sequence LVDS

1. WAIT vsync
2. BLACK_DATA=1, BLANK_DE=1, BLANK_H=1, BLANK_V=1, WAIT vsync
3. PIN_B=0, WAIT ?
4. TRISTATE_IOS=1, WAIT ?
5. PIN_A=0, WAIT ?
6. HALT=1, ASSERT_PLL_RESET=1, POWERDOWN_MACRO=1, WAIT ?

Powerup sequence TMDS

1. POWERDOWN_MACRO=0, WAIT ?
2. ASSERT_PLL_RESET=0, WAIT ?
3. TRISTATE_IOS=0, WAIT vsync
4. HALT=1, BLACK_DATA=0, BLANK_DE=0, BLANK_H=0, BLANK_V=0, WAIT 0

Powerdown sequence TMDS

1. WAIT vsync
2. BLACK_DATA=1, BLANK_DE=1, BLANK_H=1, BLANK_V=1, WAIT vsync
3. TRISTATE_IOS=1, WAIT ?
4. HALT=1, ASSERT_PLL_RESET=1, POWERDOWN_MACRO=1, WAIT ?

SOR PLL_PULLDOWN logic has been updated and allows either Sequencer controls it or a register write controls it. It can be best described as below:

```
-------------------------------------------------------------------------------------------------
|PLL_POWERDOWN  || NV_PDISP_SOR_PLL0_PULLDOWN   | NV_PDISP_SOR_PLL2_AUX0 |
NV_PDISP_SOR_SEQ_INST_PLL_PULLDOWN|
-------------------------------------------------------------------------------------------------
|SEQ Toggles it ||  DISABLE                 | PULLDOWN_ALLOW        |    1/0
|
-------------------------------------------------------------------------------------------------
|Always DISABLE ||  DISABLE                | PULLDOWN_OVERRIDE     |    Don't Care
|
-------------------------------------------------------------------------------------------------
|Always ENABLE  ||  ENABLE                 | Don't Care            |    Dont' Care
|
-------------------------------------------------------------------------------------------------
```

Note: *Register set can be accessed as an array as defined here or as individual registers as defined later.*

Usage: boot / initialization

SOR_NV_PDISP_SOR_SEQ_INST0_0: Offset: 0x22 | Byte Offset: 0x88
SOR_NV_PDISP_SOR_SEQ_INST1_0: Offset: 0x23 | Byte Offset: 0x8c

SOR_NV_PDISP_SOR_SEQ_INST2_0: Offset: 0x24 | Byte Offset: 0x90
SOR_NV_PDISP_SOR_SEQ_INST3_0: Offset: 0x25 | Byte Offset: 0x94
SOR_NV_PDISP_SOR_SEQ_INST4_0: Offset: 0x26 | Byte Offset: 0x98
SOR_NV_PDISP_SOR_SEQ_INST5_0: Offset: 0x27 | Byte Offset: 0x9c
SOR_NV_PDISP_SOR_SEQ_INST6_0: Offset: 0x28 | Byte Offset: 0xa0
SOR_NV_PDISP_SOR_SEQ_INST7_0: Offset: 0x29 | Byte Offset: 0xa4
SOR_NV_PDISP_SOR_SEQ_INST8_0: Offset: 0x2a | Byte Offset: 0xa8
SOR_NV_PDISP_SOR_SEQ_INST9_0: Offset: 0x2b | Byte Offset: 0xac
SOR_NV_PDISP_SOR_SEQ_INSTA_0: Offset: 0x2c | Byte Offset: 0xb0
SOR_NV_PDISP_SOR_SEQ_INSTB_0: Offset: 0x2d | Byte Offset: 0xb4
SOR_NV_PDISP_SOR_SEQ_INSTC_0: Offset: 0x2e | Byte Offset: 0xb8
SOR_NV_PDISP_SOR_SEQ_INSTD_0: Offset: 0x2f | Byte Offset: 0xbc
SOR_NV_PDISP_SOR_SEQ_INSTE_0: Offset: 0x30 | Byte Offset: 0xc0
SOR_NV_PDISP_SOR_SEQ_INSTF_0: Offset: 0x31 | Byte Offset: 0xc4

Read/Write: R/W | Reset: 0x01008000 (0b00000001000x00001x00xx0000000000)

Bit	Reset	Description
31	DISABLE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	NORMAL	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	NORMAL	ASSERT_PLL_RESET: 0 = NORMAL 1 = RST
28	NORMAL	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	NORMAL	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	NORMAL	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	NORMAL	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	TRISTATE	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	FALSE	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	LOW	PIN_B: 0 = LOW 1 = HIGH
21	LOW	PIN_A: 0 = LOW 1 = HIGH
19	UP	SEQUENCE: 0 = UP 1 = DOWN
18	STOP	LANE_SEQ: 0 = STOP 1 = RUN
17	NO	PDPORT: 0 = NO 1 = YES

Bit	Reset	Description
16	NO	PDPLL: 0 = NO 1 = YES
15	TRUE	HALT: 0 = FALSE 1 = TRUE
13:12	US	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

27.8.25 SOR_NV_PDISP_SOR_PWM_DIV_0

These two registers controls the optional PWM function for backlight intensity control. The PWM circuit will be driven off the buffered crystal clock. The first register (SOR_PWM_DIV) prescales the crystal clock and sets the resolution range.

DIVIDE: This field defines the period of the PWM output. Note that a value of 0 has the special meaning to disable pwm output.

pwm freq = reference clock / DIVIDE

DUTY_CYCLE: This specifies the duty cycle of PWM output. In other words, the number of cycles during pwm period (DIVIDE), hw will assert high (Hi_PERIOD).

CLK_SEL: Starting gt21x, sor pwm can be run at pixel clock. This field selects if PWM is run at crystal clock or pixel clock.

SETTING_NEW: Provides the mechanism to trigger a new configuration for both registers (SOR_PWM_DIV and SOR_PWM_CTL). When updating SOR_PWM_DIV, software must be careful to not overwrite an old value in duty cycle unless doing so is desired. Once a state change has been requested, it must be permitted to complete before attempting to trigger another one.

Examples of computation:

If Fs is the desired Frequency

DIV_DIVIDE = (reference freq / Fs); DIVIDE > 0 and integer.

DUTY_CYCLE Range: 1 to DIV_DIVIDE

To get minimum intensity: DUTY_CYCLE = 1

To get maximum intensity: DUTY_CYCLE = DIV_DIVIDE

Example 1:

For reference clock 27 MHz (Xtal)

Desired: 210 Hz (Fs)

DIV_DIVIDE = (27000000/210) = 128571.43

Round down to the 128571 causes desired Freq Fs = 210.0007 Hz

Round up to the 128572 causes desired Freq Fs = 209.999 Hz

Example 2:

For Crystal 120 MHz (pixel clock)

Desired: 50000 Hz (Fs)

DIV_DIVIDE = (120000000/50000) = 2400

Usage: boot / initialization

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:0	0x0	DIVIDE

27.8.26 SOR_NV_PDISP_SOR_PWM_CTL_0

Usage: boot / initialization / normal operation

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxx00000000000000000000000)

Bit	Reset	Description
31	DONE	SETTING_NEW: 0 = DONE 1 = PENDING 1 = TRIGGER
30	PCLK	CLKSEL: 0 = PCLK 1 = XTAL
23:0	0x0	DUTY_CYCLE

27.8.27 SOR_NV_PDISP_SOR_XBAR_CTRL_0

This register controls the xbar between the SOR and the TMDS analog macro. There are 2 links per SOR and 4/5 TMDS channels per link. Within the same link, the xbar are fully connected (i.e. each output channel can be connected to any channel). The xbar output can also be driven to zero by setting XSEL_* to XSEL_*_ZERO. Also, the two link can be swapped by setting LINK_SWAP to one. This xbar can also be bypassed by setting BYPASS to one.

Usage: boot / initialization

Offset: 0x4a | Byte Offset: 0x128 | Read/Write: R/W | Reset: 0x8d111a23 (0b10001101000100010001101000100011)

Bit	Reset	Description
31:29	0x4	LINK1_XSEL_4: 7 = ZERO
28:26	0x3	LINK1_XSEL_3: 7 = ZERO
25:23	0x2	LINK1_XSEL_2: 7 = ZERO
22:20	0x1	LINK1_XSEL_1: 7 = ZERO
19:17	0x0	LINK1_XSEL_0: 7 = ZERO
16:14	0x4	LINK0_XSEL_4: 7 = ZERO
13:11	0x3	LINK0_XSEL_3: 7 = ZERO
10:8	0x2	LINK0_XSEL_2: 7 = ZERO
7:5	0x1	LINK0_XSEL_1: 7 = ZERO
4:2	0x0	LINK0_XSEL_0: 7 = ZERO
1	0x1	LINK_SWAP
0	0x1	BYPASS

27.8.28 SOR_NV_PDISP_SOR_XBAR_POL_0

This register controls the polarity of the channels going into the xbar. Each input channel can be inverted individually by setting the corresponding bit to one.

Usage: boot / initialization

Offset: 0x4b | Byte Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	NORMAL	POL_LINK1_4: 0 = NORMAL 1 = INVERT
8	NORMAL	POL_LINK1_3: 0 = NORMAL 1 = INVERT
7	NORMAL	POL_LINK1_2: 0 = NORMAL 1 = INVERT
6	NORMAL	POL_LINK1_1: 0 = NORMAL 1 = INVERT
5	NORMAL	POL_LINK1_0: 0 = NORMAL 1 = INVERT
4	NORMAL	POL_LINK0_4: 0 = NORMAL 1 = INVERT
3	NORMAL	POL_LINK0_3: 0 = NORMAL 1 = INVERT
2	NORMAL	POL_LINK0_2: 0 = NORMAL 1 = INVERT
1	NORMAL	POL_LINK0_1: 0 = NORMAL 1 = INVERT
0	NORMAL	POL_LINK0_0: 0 = NORMAL 1 = INVERT

27.8.29 SOR_NV_PDISP_SOR_DP_LINKCTL0_0

PROGRAMMING NOTE:

The SOR_DP registers are doubly indexed. The first index selects the SOR, the second index selects port within the SOR. Some DP SORs will support DPA port and DPB port. The capability bits for each SOR will determine what ports exist.

The field NV_PDISP_SOR_DP_LINKCTL_ENABLE is used to enable a specific port on a given OR.

SOR_DP_LINKCTL

ENABLE

- _YES The current DP port is enabled and data will be driven to the associated pins
- _NO The current DP port is not enabled and the port will be powered down.

Each DP SOR has two DP ports, but only one can be active at any given time. Only one of the SOR_DP_LINKCTL_ENABLE fields should be set to _YES. Whichever set of SOR_DP registers has ENABLE set to YES will be the set that controls the DP hardware.

This field must be set to _YES before the first test pattern is sent out for link training. If this field is ever set to _NO, the link must be re-trained before it can be used again.

A certain number of active symbols are metered out every transfer unit to maintain a constant data rate. Since hardware has limited precision to represent the number to active symbols, an error will add up over the line width. TUSIZE is programmed to minimize the error to a value that is close to zero or can be handled by the buffer available in the hardware to sustain the slightly higher or lower data rate than the calculated value.

TUSIZE

Transfer unit size - Valid range 64 to 32

A certain number of active symbols are metered out every transfer unit to maintain a constant data rate. Since hardware has limited precision to represent the number to active symbols, an error will add up over the line width. TUSIZE is programmed to minimize the error to a value that is close to zero or can be handled by the buffer available in the hardware to sustain the slightly higher or lower data rate than the calculated value.

SYNCMODE

- _DISABLE = Link clock and stream clock asynchronous
- _ENABLE = Link clock and stream clock synchronous

This bit is used to specify whether the link clock and pixel clock are synchronous or not. This bit is included in the Main Stream Attribute data. The DP hardware treats all pixel clocks as asynchronous, so this should be left as _DISABLE.

ENHANCED_FRAME

- _DISABLE = Enhanced Framing symbol sequence for BS, SR, CPBS, and CPSR not supported.
- _ENABLE = Enhanced Framing symbol sequence for BS, SR, CPBS, and CPSR supported.

Enhanced Framing is required for HDCP over DP. This should be set to _ENABLE when the SOR is being used for DP and HDCP is enabled.

LANE_COUNT

- _ZERO = no lanes, DP disabled
- _ONE = One lane
- _TWO = Two lanes
- _FOUR = Four lanes

This field controls the lane configuration of the DP datapath. Power controls for individual lanes are in NV_PDISP_SOR_DP_PADCTL. For one-lane configuration, Lane0 is used. For 2-lane configuration, Lane0 and Lane1 are used. Source may choose any lane count as long as it does not exceed the capability of DisplayPort receiver as indicated in the receiver capability field. This field should only be changed when the SOR is asleep.

COMPLIANCE TEST PATTERN

This field is not implemented in Hardware

FORCE IDLE PATTERN

This is a diagnostics bit which when set will force the hardware to send idle pattern on all lanes enabled. By default, the DP hardware will automatically send out the idle pattern if the OR goes to sleep while the link is still trained. This field provides an additional override if needed.

- _NO - don't force idle pattern
- _YES - force idle pattern

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Reset: 0x00000100 (0b0xx0xxxxxxx00000x0xxx0x1000000x0)

Bit	Reset	Description
31	NO	FORCE_IDLEPTTRN: 0 = NO 1 = YES
28	NOPATTERN	COMPLIANCEPTTRN: 0 = NOPATTERN 1 = COLORSQARE

Bit	Reset	Description
20:16	ZERO	LANECOUNT: 0 = ZERO 1 = ONE 3 = TWO 15 = FOUR
14	DISABLE	ENHANCEDFRAME: 0 = DISABLE 1 = ENABLE
10	DISABLE	SYNCMODE: 0 = DISABLE 1 = ENABLE
8:2	0x40	TUSIZE
0	NO	ENABLE: 0 = NO 1 = YES

27.8.30 SOR_NV_PDISP_SOR_DP_LINKCTL1_0

Offset: 0x4d | Byte Offset: 0x134 | Read/Write: R/W | Reset: 0x00000100 (0b0xx0xxxxxxx00000x0xxx0x1000000x0)

Bit	Reset	Description
31	NO	FORCE_IDLEPTTRN: 0 = NO 1 = YES
28	NOPATTERN	COMPLIANCEPTTRN: 0 = NOPATTERN 1 = COLORSQARE
20:16	ZERO	LANECOUNT: 0 = ZERO 1 = ONE 3 = TWO 15 = FOUR
14	DISABLE	ENHANCEDFRAME: 0 = DISABLE 1 = ENABLE
10	DISABLE	SYNCMODE: 0 = DISABLE 1 = ENABLE
8:2	0x40	TUSIZE
0	NO	ENABLE: 0 = NO 1 = YES

27.8.31 SOR_NV_PDISP_SOR_LANE_DRIVE_CURRENT0_0

SOR_LANE_DRIVE_CURRENT

Transmitter main output drive level. Used to set transmitter main output drive level for each of the four lanes.

bit<7> is dummy

bit<6:3> is decoded into each 3.2mA unit

bit<2:0> are mapped to binary weighted units of 1.6mA, 0.8mA, 0.4mA. Each step is 400uA; the max setting=x100 0111=28mA

- x000 0000 -> 0mA

  ...

- x100 0111 -> 28mA max

  ...

- x111 1111 -> maxed out

Voltage is for single-ended output amplitude with double 50 ohm termination. Voltage is independent of termination control.

This register will be used during link training to set the voltage swing as requested by the sink device. The value to set for the drive current is dependent on the current value of the pre-emphasis in SOR_LANE_PREEMPHASIS. There is a separate LEVEL define for each valid pre-emphasis value.

Note: *When DP is active, the LANE0_DP_LANE2 field should be used for lane2 and LANE2_DP_LANE0 should be used for lane0.*

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3
23:16	0x0	LANE2_DP_LANE0
15:8	0x0	LANE1_DP_LANE1
7:0	0x0	LANE0_DP_LANE2

27.8.32 SOR_NV_PDISP_SOR_LANE_DRIVE_CURRENT1_0

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3
23:16	0x0	LANE2_DP_LANE0
15:8	0x0	LANE1_DP_LANE1
7:0	0x0	LANE0_DP_LANE2

27.8.33 SOR_NV_PDISP_SOR_LANE4_DRIVE_CURRENT0_0

SOR_LANE4_DRIVE_CURRENT

Non-DP SORs contain five total lanes; four data lanes and one clock. There was not enough room in one register to fit all five lanes. LANE4 controls the fifth lane.

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	LANE4

27.8.34 SOR_NV_PDISP_SOR_LANE4_DRIVE_CURRENT1_0

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	LANE4

27.8.35 SOR_NV_PDISP_SOR_LANE_PREEMPHASIS0_0

SOR_LANE_PREEMPHASIS

Transmitter main output pre-emphasis for each of the four lanes.

bit<7> is dummy

bit<6:3> is decoded into each 1.6mA unit

bit<2:0> is mapped to binary weighted units of 0.8mA, 0.4mA, 0.2mA. Each step is 200 µA; the maximum setting=x010 1111 = 9.4mA, if the main driver is not borrowing; the main driver has higher priority to borrow.

- x000 0000 -> 0mA

  ...

- x010 1111 -> 9.4mA max

  ...

- x111 1111 -> maxed out

This register will be used during link training to set the pre-emphasis level of each lane as requested by the sink device. The sink device may request this value to be changed during Training Pattern 1 or Training Pattern 2 during link training.

The pre-emphasis value for a given level is dependent on the drive current setting.

There is a separate LEVEL define for each valid drive current setting.

Note: *When DP is active, the LANE0_DP_LANE2 field should be used for lane2 and LANE2_DP_LANE0 should be used for lane0.*

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3
23:16	0x0	LANE2_DP_LANE0
15:8	0x0	LANE1_DP_LANE1
7:0	0x0	LANE0_DP_LANE2

27.8.36 SOR_NV_PDISP_SOR_LANE_PREEMPHASIS1_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3
23:16	0x0	LANE2_DP_LANE0
15:8	0x0	LANE1_DP_LANE1
7:0	0x0	LANE0_DP_LANE2

27.8.37 SOR_NV_PDISP_SOR_LANE4_PREEMPHASIS0_0

SOR_LANE4_PREEMPHASIS

Transmitter main output pre-emphasis for lane4 of each link.

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	LANE4

27.8.38 SOR_NV_PDISP_SOR_LANE4_PREEMPHASIS1_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	LANE4

27.8.39 SOR_NV_PDISP_SOR_POSTCURSOR0_0

SOR_LANE_POSTCURSOR

Controls Post-cursor2 amplitude for each lane. This is required during link training at HBR2 speeds.

This should be set to 0 for RBR or HBR training.

bit<7:5> is dummy

bit<4:3> is decoded into each 1.6mA unit

bit<2:0> is mapped to binary weighted units of 0.8mA, 0.4mA, 0.2mA. Each step is 200 µA; the maximum setting=xxx1 1111 = 6.2mA, if the main driver is not borrowing; the main driver has higher priority to borrow.

- xxx0 0000 -> 0mA

  ...

- xxx1 1111 -> 6.2mA max

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3
23:16	0x0	LANE2_DP_LANE0
15:8	0x0	LANE1_DP_LANE1
7:0	0x0	LANE0_DP_LANE2

27.8.40 SOR_NV_PDISP_SOR_POSTCURSOR1_0

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3_DP_LANE3
23:16	0x0	LANE2_DP_LANE0
15:8	0x0	LANE1_DP_LANE1
7:0	0x0	LANE0_DP_LANE2

27.8.41 SOR_NV_PDISP_SOR_DP_CONFIG0_0

SOR_DP_CONFIG

This register contains various other controls which affect the behavior of the DP logic.

WATERMARK

This field defines the number of symbols that the DP logic will wait for at the beginning of a line before ending blanking. This number will be based on the pixel rate, the link speed, bits per pixel, tusize and line active width.

IDLE_BEFORE_ATTACH

This field should be set to _ENABLE.

RD_RESET_VAL

When training pattern 2 begins, the running disparity of the 8B10B encoding is reset. The DP spec defines that the disparity of the first symbol sent out should be negative. By resetting the 8B10B running disparity to positive, the first symbol sent out will have a negative disparity.

- _POSITIVE: The internal running disparity is reset to positive, so the first 10b symbol sent out will have negative disparity.

- _NEGATIVE: The internal running disparity is reset to negative, so the first 10b symbol send out will have positive disparity.

This value is not expected to change unless a panel is encountered that expects a different disparity.

ACTIVESYM_COUNT, ACTIVESYM_FRAC and ACTIVESYM_POLARITY:

These three fields control the number of active symbols sent out every transfer unit. Hardware maintains a TU count which is used for metering. ACTIVESYM_COUNT is the number of symbols sent out every transfer irrespective of the tu count. Based on the ACTIVESYM_POLARITY which can be 0 or 1, we can send extra symbol for a particular TU based on the ACTIVESYM_FRAC (active fraction). If active polarity is 0 and ACTIVESYM_FRAC is say 5, hardware send an extra symbols every 5th TU. If active polarity is 1 and ACTIVESYM_FRAC is 5, hardware will send an extra symbols every TU except for the 5th TU.

For instance if the number of active symbols per TU is calculated to be 17.2, ACTIVESYM_COUNT is set to 17. Fractional value is 0.2 = 1/5 which translates to 1 symbol every 5 TUs. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 0.

For instance if the number of active symbols per TU is calculated to be 17.8, ACTIVESYM_COUNT is set to 17. Fractional value is 0.8 = 4/5 which translates to 1 symbol every TU except for the 5th TU. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 1.

ACTIVESYM_COUNT, ACTIVESYM_FRAC and ACTIVESYM_POLARITY:

These three fields control the number of active symbols sent out every transfer unit. Hardware maintains a TU count which is used for metering. ACTIVESYM_COUNT is the number of symbols sent out every transfer irrespective of the tu count. Based on the ACTIVESYM_POLARITY which can be 0 or 1, we can send extra symbol for a particular TU based on the ACTIVESYM_FRAC (active fraction). If active polarity is 0 and ACTIVESYM_FRAC is say 5, hardware send an extra symbols every 5th TU. If active polarity is 1 and ACTIVESYM_FRAC is 5, hardware will send an extra symbols every TU except for the 5th TU.

For instance if the number of active symbols per TU is calculated to be 17.2, ACTIVESYM_COUNT is set to 17. Fractional value is 0.2 = 1/5 which translates to 1 symbol every 5 TU's. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 0.

For instance if the number of active symbols per TU is calculated to be 17.8, ACTIVESYM_COUNT is set to 17. Fractional value is 0.8 = 4/5 which translates to 1 symbol every TU except for the 5th TU. This translates to ACTIVESYM_FRAC of 5 and ACTIVESYM_POLARITY of 1.

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0x94000000 (0b1xx1x1x0xxxx0000x0000000xx000000)

Bit	Reset	Description
31	NEGATIVE	RD_RESET_VAL: 0 = POSITIVE 1 = NEGATIVE
28	ENABLE	IDLE_BEFORE_ATTACH: 0 = DISABLE 1 = ENABLE
26	ENABLE	ACTIVESYM_CNTL: 0 = DISABLE 1 = ENABLE
24	NEGATIVE	ACTIVESYM_POLARITY: 0 = NEGATIVE 1 = POSITIVE
19:16	0x0	ACTIVESYM_FRAC
14:8	0x0	ACTIVESYM_COUNT
5:0	0x0	WATERMARK

27.8.42 SOR_NV_PDISP_SOR_DP_CONFIG1_0

Offset: 0x59 | Byte Offset: 0x164 | Read/Write: R/W | Reset: 0x94000000 (0b1xx1x1x0xxxx0000x0000000xx000000)

Bit	Reset	Description
31	NEGATIVE	RD_RESET_VAL: 0 = POSITIVE 1 = NEGATIVE
28	ENABLE	IDLE_BEFORE_ATTACH: 0 = DISABLE 1 = ENABLE
26	ENABLE	ACTIVESYM_CNTL: 0 = DISABLE 1 = ENABLE
24	NEGATIVE	ACTIVESYM_POLARITY: 0 = NEGATIVE 1 = POSITIVE
19:16	0x0	ACTIVESYM_FRAC
14:8	0x0	ACTIVESYM_COUNT
5:0	0x0	WATERMARK

27.8.43 SOR_NV_PDISP_SOR_DP_MN0_0

SOR_DP_MN

This register contains controls to tweak the values used for M and N in the Main Stream Attribute data. These controls are for debugging purposes and possible workarounds. See Section 2.3.3 of the DisplayPort specification for more detailed descriptions of M and N.

N_VAL

This is the value that will be used for calculating M. This value is not used in the Main Stream Attributes. Main Stream attributes always send 2^15 for Nvid. This should not need to be programmed beyond its default value. If this value needs to be changed, it should be done while the OR is detached. Otherwise, the calculated M value will be incorrect for some time, or until the next mode switch.

M_DELTA: This defines an offset that will be used to modify the calculated M value. This should not need to be programmed beyond its default value.

M_MOD: Describes how the M_DELTA field should be applied to the M value.

- _NONE: The M_DELTA is ignored and not used.
- _INC: The M_DELTA value is added to M.
- _DEC: The M_DELTA value is subtracted from M.

This should not need to be programmed beyond its default value.

Offset: 0x5a | Byte Offset: 0x168 | Read/Write: R/W | Reset: 0x00008000 (0b00xx00000000000010000000000000000)

Bit	Reset	Description
31:30	NONE	M_MOD: 0 = NONE 1 = INC 2 = DEC
27:24	0x0	M_DELTA
23:0	0x8000	N_VAL

27.8.44 SOR_NV_PDISP_SOR_DP_MN1_0

Offset: 0x5b | Byte Offset: 0x16c | Read/Write: R/W | Reset: 0x00008000 (0b00xx00000000000010000000000000000)

Bit	Reset	Description
31:30	NONE	M_MOD: 0 = NONE 1 = INC 2 = DEC
27:24	0x0	M_DELTA
23:0	0x8000	N_VAL

27.8.45 SOR_NV_PDISP_SOR_DP_PADCTL0_0

SOR_DP_PADCTL

This register directly controls the DP pads. Some values in this register are only valid in DP mode when NV_PDISP_SOR_DP_LINKCTL_ENABLE == _YES.

PD_TXD_

Individual controls to power down the lanes. NV_PDISP_SOR_DP_LINKCTL_LANE_COUNT sets the number of lanes that are active in the DP datapath. These fields control power to the macro pins. When reducing the number of active lanes, the inactive lanes should be powered down before the lane configuration is changed. See Section 5.1.2 of the DP Specification.

If a lane is powered down, link training will need to be done before that lane can be used again.

- _YES: Force the lane to power down
- _NO: Enable power to the lane. Do not power down.

Only valid in DP mode when NV_PDISP_SOR_DP_LINKCTL_ENABLE == _YES.

COMMONMODE_TXD_

This is used to force the lanes to output the common mode voltage. This is required before link training begins. Section 3.5.3.3 of the DP specification dictates that the lanes must be precharged to the common mode voltage for at least 10 microseconds before beginning link training. Please see the DP specification for more information:

> ***Note:*** *When DP is active, the TXD_0_DP_TXD_2 field should be used for lane2 and TXD_2_DP_TXD_0 should be used for lane0.*

- _DISABLE: Normal operation
- _ENABLE: Force the lanes to output the common mode voltage

TX_PU_VALUE: TX pull-up current source drive

- x000xxxx -> 0mA
- x001xxxx -> 1mA
- x010xxxx -> 2mA
- x011xxxx -> 3mA
- x100xxxx -> 4mA
- x101xxxx -> 5mA
- x110xxxx -> 6mA

TX_PU

- _ENABLE: Pull-up current sources enabled

- _DISABLE: Pull-up current sources disabled

Currently it is unknown if this will be needed.

VCMMODE

VCM pin mode select

- _TRISTATE:tristate (default)
- _TEST_MUX:under test-mux control
- _WEAK_PULLDOWN:0 V weak pull-down
- _STRONG_PULLDOWN: 0V strong pull-down

This is a test mode pin, it does not need to be programmed by software.

REG_CTRL: Internal regulator control. These are spare input bits to the analog macro that have no defined function currently.

PAD_CAL_PD

Powerdown control for the pad calibration logic.

- _POWERDOWN: Power down
- _POWERUP:Normal operation

VCO_2X: This is a VCO startup diagnostics control input to the DP analog macro.

- _ENABLE: forced VCO oscillation mode
- _DISABLE: normal VCO operation

VCOLIMIT_SEL:

- When vcolimit_sel=0, the clamp counter counts 128 cycles <-> 474 ns @270 MHz;
- When vcolimit_sel=1, the clamp counter counts 32 cycles <-> 118 ns @ 270 MHz.

LOADADJ_SYNC_EN:

- When loadadj_sync_en=0,the loadadj-calibration code is reset to 0000;
- When loadadj_sync_en=1,the loadadj-calibration code is swept and the best value is used.

Used during LOADADJ calibration. Must be set to 0 and then to 1 to trigger the calibration sequence.

LOADADJ_BYPN:

- When loadadj_bypn=0, the loadadj-calibration block is bypassed;
- When loadadj_bypn=1, the loadadj-calibration block is being used.

Used during LOADADJ calibration to select the auto-cal value instead of the register value.

VCOCALIB_ENB:

- When vcocalib_enb=0,enable vco-calibration, sweep the vcogain code and the best value is used.
- When vcocalib_enb=1, reset the vco-calibration block.

Used during VCO Gain Calibration. Must be set to 1 and then back to 0 to trigger the calibration sequence.

VCOCALIB_OVERWRB:

- When vcocalib_overwrb=0, overwrite/disable the vco-calibration block;
- When vcocalib_overwrb=1, enable the vco-calibration block.

Used during VCO Gain Calibration.

VCOCALIB_TS0:

- When vcocalib_ts0=0, vco-calibration block counts 96 cycles <-> 355 ns @270 MHz
- When vcocalib_ts0=1, vco-calibration block counts 128 cycles <-> 474 ns @270 MHz

VCOLIMIT_DISABLE:

- When vcolimit_disable=0, enable the VCO-protection clamp;
- When vcolimit_disable=1, disable the VCO-protection clamp.

Offset: 0x5c | Byte Offset: 0x170 | Read/Write: R/W | Reset: 0x00800000 (0b00000000100000000000000000000000)

Bit	Reset	Description
31	DISABLE	VCOLIMIT_DISABLE: 0 = DISABLE 1 = ENABLE
30	0x0	VCOCALIB_TS0
29	DISABLE	VCOCALIB_OVERWRB: 0 = DISABLE 1 = ENABLE
28	0x0	VCOCALIB_ENB
27	DISABLE	LOADADJ_BYPN: 0 = DISABLE 1 = ENABLE
26	0x0	LOADADJ_SYNC_EN
25	0x0	VCOLIMIT_SEL
24	DISABLE	VCO_2X: 0 = DISABLE 1 = ENABLE
23	POWERDOWN	PAD_CAL_PD: 0 = POWERUP 1 = POWERDOWN
22	DISABLE	TX_PU: 0 = DISABLE 1 = ENABLE
21:20	0x0	REG_CTRL
19:16	TRISTATE	VCMMODE: 0 = TRISTATE 1 = TEST_MUX 2 = WEAK_PULLDOWN 4 = STRONG_PULLDOWN
15:8	0x0	TX_PU_VALUE
7	DISABLE	COMMONMODE_TXD_3_DP_TXD_3: 0 = DISABLE 1 = ENABLE
6	DISABLE	COMMONMODE_TXD_2_DP_TXD_0: 0 = DISABLE 1 = ENABLE
5	DISABLE	COMMONMODE_TXD_1_DP_TXD_1: 0 = DISABLE 1 = ENABLE
4	DISABLE	COMMONMODE_TXD_0_DP_TXD_2: 0 = DISABLE 1 = ENABLE
3	YES	PD_TXD_3: 0 = YES 1 = NO
2	YES	PD_TXD_0: 0 = YES 1 = NO
1	YES	PD_TXD_1: 0 = YES 1 = NO

Bit	Reset	Description
0	YES	PD_TXD_2: 0 = YES 1 = NO

27.8.46 SOR_NV_PDISP_SOR_DP_PADCTL1_0

Offset: 0x5d | Byte Offset: 0x174 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	DISABLE	VCOLIMIT_DISABLE: 0 = DISABLE 1 = ENABLE
30	0x0	VCOCALIB_TS0
29	DISABLE	VCOCALIB_OVERWRB: 0 = DISABLE 1 = ENABLE
28	0x0	VCOCALIB_ENB
27	DISABLE	LOADADJ_BYPN: 0 = DISABLE 1 = ENABLE
26	0x0	LOADADJ_SYNC_EN
25	0x0	VCOLIMIT_SEL
24	DISABLE	VCO_2X: 0 = DISABLE 1 = ENABLE
23	POWERUP	PAD_CAL_PD: 0 = POWERUP 1 = POWERDOWN
22	DISABLE	TX_PU: 0 = DISABLE 1 = ENABLE
21:20	0x0	REG_CTRL
19:16	TRISTATE	VCMMODE: 0 = TRISTATE 1 = TEST_MUX 2 = WEAK_PULLDOWN 4 = STRONG_PULLDOWN
15:8	0x0	TX_PU_VALUE
7	DISABLE	COMMONMODE_TXD_3_DP_TXD_3: 0 = DISABLE 1 = ENABLE
6	DISABLE	COMMONMODE_TXD_2_DP_TXD_0: 0 = DISABLE 1 = ENABLE
5	DISABLE	COMMONMODE_TXD_1_DP_TXD_1: 0 = DISABLE 1 = ENABLE
4	DISABLE	COMMONMODE_TXD_0_DP_TXD_2: 0 = DISABLE 1 = ENABLE
3	YES	PD_TXD_3: 0 = YES 1 = NO
2	YES	PD_TXD_0: 0 = YES 1 = NO
1	YES	PD_TXD_1: 0 = YES 1 = NO

Bit	Reset	Description
0	YES	PD_TXD_2: 0 = YES 1 = NO

27.8.47 SOR_NV_PDISP_SOR_DP_SPARE0_0

NV_PDISP_SOR_DP_SPARE

Reserved for future features

SEQ_ENABLE

Used to enable the sequencer in DP mode. This needs to be set for eDP where we need to run sequencer to control backlight.

PANEL

Specify whether the DP panel is external or internal (notebook panel)

- EXTERNAL: An external DP panel is connected, normal operation
- INTERNAL: This port is connected to an internal panel. In this mode, the upstream status will report an internal link, and the scrambler will reset to 0xFFFE instead of 0xFFFF

SOR_CLK_SEL

This is used for SOR testbench. Specify whether to use a safe clock or the macro clock as the SOR clock.

- SAFE_SORCLK: Use the safe clock as the SOR clock
- MACRO_SORCLK: Use the macro clock as the SOR clock

DISP_VIDEO_PREAMBLE_CYA

This is used to select between video preamble from display versus the locally generated video preamble signal

- DISABLE: Use locally generated video preamble
- ENABLE: Use video preamble from display (To be used for legacy support of HDMI programming)

Offset: 0x60 | Byte Offset: 0x180 | Read/Write: R/W | Reset: 0x00000002 (0b00000000000000000000000000000010)

Bit	Reset	Description
31	DISABLE	SOR_PSR_DIABLE_CYA: 0 = DISABLE 1 = ENABLE
30:4	0x0	REG
3	DISABLE	DISP_VIDEO_PREAMBLE_CYA: 0 = DISABLE 1 = ENABLE
2	SAFE_SORCLK	SOR_CLK_SEL: 0 = SAFE_SORCLK 1 = MACRO_SORCLK
1	INTERNAL	PANEL: 0 = EXTERNAL 1 = INTERNAL
0	NO	SEQ_ENABLE: 0 = NO 1 = YES

27.8.48 SOR_NV_PDISP_SOR_DP_SPARE1_0

Offset: 0x61 | Byte Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:3	0x0	REG

Bit	Reset	Description
2	SAFE_SORCLK	SOR_CLK_SEL: 0 = SAFE_SORCLK 1 = MACRO_SORCLK
1	EXTERNAL	PANEL: 0 = EXTERNAL 1 = INTERNAL
0	NO	SEQ_ENABLE: 0 = NO 1 = YES

27.8.49 SOR_NV_PDISP_SOR_DP_AUDIO_CTRL_0

SOR_DP_AUDIO_CTRL

ENABLE

This is the global enable/disable field for Audio over DisplayPort. When set to _NO, no audio stream packets, audio timestamp packets, or audio infoframes will be sent. When set to _YES, audio stream packets will be sent whenever audio arrives, and timestamp and infoframe packets will be sent once per frame during hblank.

MUTE

This field controls the AudioMute_Flag in the VB-ID. When muted, audio samples sent to the sink device are not played. This is an override to the audio mute signal coming from the Azalia Codec.

- DISABLE: The mute is not asserted, audio will be audible
- ENABLE: The mute is asserted. The sink device should not play any audio samples that are sent. If SW changes MUTE from ENABLE to AUTO, hardware will take control of unmuting after the new audio infoframe and audio timestamp have been sent.
- AUTO: The mute signal is controlled by hardware. Audio is muted during audio format or audio timing change and unmuted after the new audio infoframe and audio timestamp have been sent.

AUDIO INFOFRAME HEADER_OVERRIDE

HB1-HB3 are fixed values as defined by the DisplayPort specification. This field will let the values in NV_PDISP_HDMI_AUDIO_INFOFRAME_HEADER override the default values. This can be used for Debugging purposes and can be left at DISABLE for production.

- DISABLE: Use the default values of HB0=0x00, HB1=0x84, HB2=0x1B, HB3=0x44
- ENABLE: Replace the contents of the Audio Infoframe Header with the values in NV_PDISP_HDMI_AUDIO_INFOFRAME_HEADER

GENERIC_INFOFRAME_ENABLE

This field allows software to send infoframes (AVI, etc.) other than audio infoframe. When enabled NV_PDISP_SOR_DP_GENERIC_INFOFRAME_HEADER, NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK* priv regs is used as infoframe header and packet data. The infoframe is sent once per vertical blanking period.

PACKETID

This field can be left at 0.

CT_SELECT (Coding Type), CC_SELECT (Channel Count), SF_SELECT (Sampling Frequency), SS_SELECT (Sample Size), CA_SELECT (Channel/Speaker Allocation)

These fields control the values of several fields in the Audio Infoframe. The Azalia codec will detect these values, and can automatically place the correct values in the Infoframe.

- SW: The Audio Infoframe will contain the value from NV_PDISP_HDMI_AUDIO_INFOFRAME_SUBPACK0*
- HW: The Audio Infoframe will contain the value set by the Azalia codec.

- NEW_SETTINGS

All the fields/settings in this register will not take effect until NEW_SETTINGS is set to _TRIGGER. When the new settings are active, this field will automatically be set to _DONE. If hardware is sending an audio packet when this is set to trigger, then it will wait for the completion of the packet before the new settings are activated. Hence the maximum delay from TRIGGER to DONE will be time to complete an audio packet (56* (1/162 * 10^6) = 0.35 µs).

Offset: 0x62 | Byte Offset: 0x188 | Read/Write: R/W | Reset: 0x00Xf0001 (0b0xxxxxxxxxx1111100000000000xx00x1)

Bit	R/W	Reset	Description
31	RW	DONE	NEW_SETTINGS:w1c 0 = DONE 1 = PENDING 1 = TRIGGER
21	RO	X	MUTE_STATUS: 0 = DISABLE 1 = ENABLE
20	RW	HW	CA_SELECT: 0 = SW 1 = HW
19	RW	HW	SS_SELECT: 0 = SW 1 = HW
18	RW	HW	SF_SELECT: 0 = SW 1 = HW
17	RW	HW	CC_SELECT: 0 = SW 1 = HW
16	RW	HW	CT_SELECT: 0 = SW 1 = HW
15:8	RW	0x0	PACKET_ID
7	RW	NO	GENERIC_INFOFRAME_ENABLE: 0 = NO 1 = YES
6	RW	DISABLE	INFOFRAME_HEADER_OVERRIDE: 0 = DISABLE 1 = ENABLE
3:2	RW	AUTO	MUTE: 0 = AUTO 1 = DISABLE 2 = ENABLE
0	RW	YES	ENABLE: 0 = NO 1 = YES

27.8.50 SOR_NV_PDISP_SOR_DP_AUDIO_HBLANK_SYMBOLS_0

SOR_DP_AUDIO_HBLANK_SYMBOLS

The packet generation logic needs to know the length of the hblank period. If there is no room in the current hblank for a new packet, it will be delayed until the next blanking period. This field should be programmed during the second Supervisor interrupt based on the new raster dimensions.

Y = 12/number of lanes

number of symbols/hblank = ((SetRasterBlankEnd.X + SetRasterSize.Width - SetRasterBlankStart.X - 7) * link_clk / pclk) - 3*enhanced_framing - Y

Y accounts for the time required to send BS, VBID, Mvid, Maud.

The value 7 is subtracted to account for reduction in hblank interval in the link clock domain due to uncertainties arising due to pixels crossing async boundary from pixel clock (rg_clk) to link clock (or_clk). Since we use two async FIFOs to transfer pixels from rg_clk to orclk, the uncertainties are effectively doubled. The value has been experimentally determined.

The following formulas can be used to calculate the maximum audio sampling rate that can be supported by DisplayPort given the current raster dimensions. DisplayPort has much more bandwidth during blanking periods than HDMI has, so hblank size is less of an issue.

- number of free symbols/hblank = ((SetRasterBlankEnd.X + SetRasterSize.Width - SetRasterBlankStart.X - 7) * link_clk / pclk - 2 - (3*enhanced_framing)) * #lanes - 12
- Size of a packet for 2ch audio = 20 symbols (up to 2 samples)
- Size of a packet for 8ch audio = 40 symbols
- Size of an audio packet header plus control symbols = 2*#lanes + 8 symbols (assuming < 32 samples per line)
- number of packets/hblank for 2ch audio = Floor ((number of free symbols/hblank - (2*#lanes + 8) / 20)
- number of packets/hblank for 8ch audio = Floor ((number of free symbols/hblank - (2*#lanes + 8) / 40)

Maximum audio sample rate possible:

- number of audio samples/line = SetRasterSize.Width * audio_fs / pclk
- number of audio packets needed for 2ch audio = Ceiling(SetRasterSize.Width * audio_fs / (pclk*2))
- number of audio packets needed for 3-8ch audio = SetRasterSize.Width * audio_fs / pclk

If the number of audio packets needed > number of packets/hblank, then that audio frequency is not supported.

Offset: 0x63 | Byte Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16:0	0x0	VALUE

27.8.51 SOR_NV_PDISP_SOR_DP_AUDIO_VBLANK_SYMBOLS_0

SOR_DP_AUDIO_VBLANK_SYMBOLS

The packet generation logic needs to know the length of a line during VBLANK. If there is no room in the current line for a new packet, it will be delayed until the next line. This field should be programmed during the second Supervisor interrupt based on the new raster dimensions.

Y accounts for the time required to send main stream attributes.

25 accounts for reduction of the vblank interval in the link clock domain due to active region transmission spilling on to the vblank region. Note this happen only on the first vblank line while transitioning from active to blanking. This value is experimentally determined. This might need some adjustment if NV_PDISP_SOR_AUDIO_DEBUG_BLANK_COUNT_ERROR is triggered. Note that this value will not exceed TUSIZE.

Y = (36/number of lanes) + 3;

number of symbols/line in vblank = ((SetRasterBlankStart.X - SetRasterBlankEnd.X - 25) * link_clk / pclk) - Y - 1

Offset: 0x64 | Byte Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:0	0x0	VALUE

27.8.52 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_HEADER_0

SOR_DP_GENERIC_INFOFRAME_HEADER

This register should be written with the value of the DP InfoFrame header (infoframe other than audio infoframe).

Offset: 0x65 | Byte Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	HB3
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

27.8.53 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK0_0

SOR_DP_GENERIC_INFOFRAME_SUBPACK0

This register should be written with the bytes of the DP Infoframe packet data i.e., DB0- DB27 in Figure 2-21 of the DP specification, v1.1a.

Offset: 0x66 | Byte Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB3
23:16	0x0	DB2
15:8	0x0	DB1
7:0	0x0	DB0

27.8.54 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK1_0

Offset: 0x67 | Byte Offset: 0x19c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB7
23:16	0x0	DB6
15:8	0x0	DB5
7:0	0x0	DB4

27.8.55 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK2_0

Offset: 0x68 | Byte Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB11
23:16	0x0	DB10
15:8	0x0	DB9
7:0	0x0	DB8

27.8.56 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK3_0

Offset: 0x69 | Byte Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB15
23:16	0x0	DB14
15:8	0x0	DB13
7:0	0x0	DB12

27.8.57 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK4_0

Offset: 0x6a | Byte Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB19
23:16	0x0	DB18
15:8	0x0	DB17
7:0	0x0	DB16

27.8.58 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK5_0

Offset: 0x6b | Byte Offset: 0x1ac | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB23
23:16	0x0	DB22
15:8	0x0	DB21
7:0	0x0	DB20

27.8.59 SOR_NV_PDISP_SOR_DP_GENERIC_INFOFRAME_SUBPACK6_0

Offset: 0x6c | Byte Offset: 0x1b0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DB27
23:16	0x0	DB26
15:8	0x0	DB25
7:0	0x0	DB24

27.8.60 SOR_NV_PDISP_SOR_DP_TPG_0

This register is used to control the training patterns needed during link training. This should be set after the SOR clock has been set up and the lanes have been precharged.

There is one field per lane. If the same test pattern is to be sent on all lanes, they must all be switched to that pattern at the same time in order to stay in sync

Link Training Pattern Setting

- _NOPATTERN Training not in progress (or disabled)
- _TRAINING1 Training Pattern 1
- _TRAINING2 Training Pattern 2
- _TRAINING3 Training Pattern 3
- _D102 D10.2 test pattern (unscrambled) transmitted (same as Training Pattern 1)
- _SBLEERRATE Symbol Error Rate measurement pattern transmitted
- _PRBS7 PRBS7 transmitted
- _CSTM Custom 80 bit test pattern defined in NV_PDISP_SOR_DP_LQ_CSTM
- _HBR2_COMPLIANCE HBR2 Compliance Eye pattern. Repetition of scrambled 0s before 8b10b encoding. Sends SR periodically as defined in NV_PDISP_SOR_DP_TPG_CONFIG_HBR2_COMPLIANCE_PERIOD

  This is used for HBR2 electrical compliance.

SCRAMBLEREN

- _DISABLE - DisplayPort transmitter disables scrambler
- _ENABLE_GALIOS - DisplayPort transmitter scrambles data symbols before transmission

CHANNEL_CODING

This bit set to _ENABLE when DisplayPort receiver supports the Main Link channel coding specification as specified in ANSI X3.230-1994, clause 11. This is reserved for future expansion.

See Section 27.8.60: SOR_NV_PDISP_SOR_DP_TPG_0 for more information about the different test patterns. For convenience, the table below specifies the required SCRAMBLEREN and CHANNELCODING values for each pattern.

```
                | Requires        | Requres
Pattern         | Channel Coding? | Scrambling?
----------------+-----------------+------------
TRAINING1       | Yes             | No
TRAINING2       | Yes             | No
TRAINING3       | Yes             | No
D102            | Yes             | No
SBLERRATE       | Yes             | Yes
PRBS7           | No              | No
CSTM            | No              | No
HBR2_COMPLIANCE | Yes             | Yes
```

Offset: 0x6d | Byte Offset: 0x1b4 | Read/Write: R/W | Reset: 0x50505050 (0bx1010000x1010000x1010000x1010000)

Bit	Reset	Description
30	ENABLE	LANE3_CHANNELCODING: 0 = DISABLE 1 = ENABLE
29:28	ENABLE_GALIOS	LANE3_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS
27:24	NOPATTERN	LANE3_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2 3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE
22	ENABLE	LANE2_CHANNELCODING: 0 = DISABLE 1 = ENABLE
21:20	ENABLE_GALIOS	LANE2_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS
19:16	NOPATTERN	LANE2_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2 3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE
14	ENABLE	LANE1_CHANNELCODING: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13:12	ENABLE_GALIOS	LANE1_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS
11:8	NOPATTERN	LANE1_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2 3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE
6	ENABLE	LANE0_CHANNELCODING: 0 = DISABLE 1 = ENABLE
5:4	ENABLE_GALIOS	LANE0_SCRAMBLEREN: 0 = DISABLE 1 = ENABLE_GALIOS
3:0	NOPATTERN	LANE0_PATTERN: 0 = NOPATTERN 1 = TRAINING1 2 = TRAINING2 3 = TRAINING3 4 = D102 5 = SBLERRRATE 6 = PRBS7 7 = CSTM 8 = HBR2_COMPLIANCE

27.8.61 SOR_NV_PDISP_SOR_DP_TPG_CONFIG_0

NV_PDISP_SOR_DP_TPG_CONFIG

Additional controls for the DP Test Pattern Generator.

HBR2_COMPLIANCE_PERIOD

The HBR2 compliance pattern is a series of 0s that are scrambled and 8b10b encoded. Periodically, the Scrambler Reset Sequence is sent to keep the source and sink in sync. This field controls how many total symbols there are between the start of a new Scrambler Reset Sequence. This will be defined by a DPCD register on the sink test equipment.

Offset: 0x6e | Byte Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16:0	0x0	HBR2_COMPLIANCE_PERIOD

27.8.62 SOR_NV_PDISP_SOR_DP_LQ_CSTM0_0

NV_PDISP_SOR_DP_LQ_CSTM

These registers are used to program a custom 80-bit test pattern used for link quality purposes in DP1.2. When NV_PDISP_SOR_DP_TPG is set to CSTM, the 80 bit pattern in these registers is send repetitively on the link. NV_PDISP_SOR_DP_TPG_LANE*_CHANNELCODING and NV_PDISP_SOR_DP_TPG_LANE*_SCRAMBLEREN should be set to _DISABLE in order to send out this pattern unaltered. For the HBR2 Compliance Eye Pattern, these three registers should all be set to 0, while NV_PDISP_SOR_DP_TPG_LANE*_CHANNELCODING and NV_PDISP_SOR_DP_TPG_LANE*_SCRAMBLEREN are set to _ENABLE.

- SOR_DP_LQ_CSTM0 contains bits 31:0
- SOR_DP_LQ_CSTM1 contains bits 63:32
- SOR_DP_LQ_CSTM2 contains bits 79:64

Offset: 0x6f | Byte Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYM

27.8.63 SOR_NV_PDISP_SOR_DP_LQ_CSTM1_0

Offset: 0x70 | Byte Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYM

27.8.64 SOR_NV_PDISP_SOR_DP_LQ_CSTM2_0

Offset: 0x71 | Byte Offset: 0x1c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYM

27.8.65 SOR_NV_PDISP_SOR_PLL4_0

Offset: 0x72 | Byte Offset: 0x1c8 | Read/Write: R/W | Reset: 0x00X00000 (0bxx00000000xxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
29:24	RW	0x0	SETUP_LCKDET
23	RW	DISABLE	LOCK_OVERRIDE: 0 = DISABLE 1 = ENABLE
22	RW	ENABLE	ENABLE_LCKDET: 0 = ENABLE 1 = DISABLE
21	RO	X	LOCKDET

27.8.66 SOR_NV_PDISP_SOR_DP_PADCTL2_0

Offset: 0x73 | Byte Offset: 0x1cc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xxx0)

Bit	Reset	Description
31:24	0x0	SPAREPLL
23:20	0x0	SPARE4
19:16	0x0	SPARE3
15:12	0x0	SPARE2
11:8	0x0	SPARE1
7:4	0x0	SPARE0
0	0x0	REG_BYPASS

27.8.67 SOR_NV_PDISP_SOR_DP_PADCTL3_0

Offset: 0x74 | Byte Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	TX_PATTERN_GEN

27.8.68 SOR_NV_PDISP_SOR_DP_HDCP_AN_MSB_0

HDCP control registers in the pipeline.

HDCP AN MSB REGISTER

The AN_MSB register holds the 32 most significant bits of the link encryption session random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x75 | Byte Offset: 0x1d4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.69 SOR_NV_PDISP_SOR_DP_HDCP_AN_LSB_0

HDCP AN LSB REGISTER

The AN_LSB register holds the 32 least significant bits of the link encryption session random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x76 | Byte Offset: 0x1d8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.70 SOR_NV_PDISP_SOR_DP_HDCP_AKSV_MSB_0

HDCP AKSV MSB REGISTER

The AKSV_MSB register holds the 8 most significant bits of the transmitter's downstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x77 | Byte Offset: 0x1dc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: 0 = ZERO

27.8.71 SOR_NV_PDISP_SOR_DP_HDCP_AKSV_LSB_0

HDCP AKSV LSB REGISTER

The AKSV_LSB register holds the 32 least significant bits of the transmitter's downstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x78 | Byte Offset: 0x1e0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.72 SOR_NV_PDISP_SOR_DP_HDCP_BKSV_MSB_0

HDCP BKSV MSB REGISTER

The BKSV_MSB register holds the 8 most significant bits of the receiver's key selection vector (KSV), as well as the receiver's REPEATER bit. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x79 | Byte Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31	0x0	REPEATER
7:0	ZERO	VALUE: 0 = ZERO

27.8.73 SOR_NV_PDISP_SOR_DP_HDCP_BKSV_LSB_0

HDCP BKSV LSB REGISTER

The BKSV_LSB register holds the 32 least significant bits of the receiver's key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x7a | Byte Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	ZERO	VALUE: 0 = ZERO

27.8.74 SOR_NV_PDISP_SOR_DP_HDCP_CTRL_0

HDCP CTRL REGISTER

The CTRL register is used to facilitate synchronization between the display driver and hardware.

- RUN: Set by the driver to start or stop the downstream protocol
- CRYPT: Set by the driver to turn encryption on.
- CRYPT_STATUS: This read only field reflects the encryption status of the link. Before entering DP flush mode, software has to disable encryption by setting RUN to NO. Encryption is turned off at the next scrambler reset. Hence software needs to wait until encryption is disabled by polling on CRYPT_STATUS to be _DISABLED before turning ON flush mode.
- AN: A bit set by hardware when a valid An value has been generated
- R0: A bit set by hardware when Km, Ks, M0, and R0 have been calculated
- SROM_EN: Set by hardware while the HDCP SROM is in use, cleared when not in use
- SROM_ERR: Set by hardware when an error occurs while using the HDCP SROM
- UPSTREAM: When the SOR is unattached state,
  - If this bit is set to _DP_MODE, SPRIME and MPRIME calculations will use AN,BKSV and M0 values from the DP HDCP block.
  - If this bit is set to _TMDS_MODE, SPRIME and MPRIME calculations will use AN,BKSV and M0 values from the TMDS HDCP block.

  If the SOR is attached to any HEAD, UPSTREAM SPRIME and MPRIME calculations would use AN,BKSV and M0 values from the HDCP block defined as below.

```
SOR ACTIVE PROTOCOL                                                    HDCP BLOCK
=======================================================================
SOR_SET_CONTROL_PROTOCOL_LVDS_CUSTOM              TMDS
SOR_SET_CONTROL_PROTOCOL_SINGLE_TMDS_A           TMDS
SOR_SET_CONTROL_PROTOCOL_SINGLE_TMDS_B           TMDS
SOR_SET_CONTROL_PROTOCOL_SINGLE_TMDS_AB        TMDS
SOR_SET_CONTROL_PROTOCOL_DUAL_SINGLE_TMDS     TMDS
SOR_SET_CONTROL_PROTOCOL_DUAL_TMDS                   TMDS
SOR_SET_CONTROL_PROTOCOL_DP_A                                 DP
SOR_SET_CONTROL_PROTOCOL_DP_B                                 DP
SOR_SET_CONTROL_PROTOCOL_CUSTOM                          TMDS
```

Usage: normal operation

Offset: 0x7b | Byte Offset: 0x1ec | Read/Write: R/W | Reset: 0x0000XX00 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxx0000)

Bit	R/W	Reset	Description
15	RW	0x0	UPSTREAM: 0 = TMDS_MODE 1 = DP_MODE
13	RO	X	SROM_ERR: 0 = NOERROR 1 = ERROR
12	RO	X	SROM_EN: 0 = DISABLED 1 = ENABLED
11	RO	X	MPRIME: 0 = INVALID 1 = VALID
10	RO	X	SPRIME: 0 = INVALID 1 = VALID
9	RO	X	R0: 0 = INVALID 1 = VALID
8	RO	X	AN: 0 = INVALID 1 = VALID
3	RW	0x0	ONEONE: 0 = DISABLED 1 = ENABLED
2	RW	0x0	DUAL_LINK_EN: 0 = DISABLED 1 = ENABLED
1	RW	0x0	CRYPT: 0 = DISABLED 1 = ENABLED
0	RW	0x0	RUN: 0 = NO 1 = YES

27.8.75 SOR_NV_PDISP_SOR_DP_HDCP_RI_0

HDCP RI REGISTER

The RI register holds the 16-bit link integrity check value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x7c | Byte Offset: 0x1f0 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	VALUE: 0 = ZERO

27.8.76 SOR_NV_PDISP_SOR_DP_HDCP_EMU1_0

Offset: 0x7e | Byte Offset: 0x1f8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	ZERO	VALUE: 0 = ZERO

27.8.77 SOR_NV_PDISP_SOR_DP_HDCP_CYA_0

HDCP Diagnostic Register

Usage: normal operation

Offset: 0x7f | Byte Offset: 0x1fc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

27.8.78 SOR_NV_PDISP_SOR_TMDS_HDCP_AN_MSB_0

HDCP control registers in the pipeline.

HDCP AN MSB REGISTER

The AN_MSB register holds the 32 most significant bits of the link encryption session random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x80 | Byte Offset: 0x200 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.79 SOR_NV_PDISP_SOR_TMDS_HDCP_AN_LSB_0

HDCP AN LSB REGISTER

The AN_LSB register holds the 32 least significant bits of the link encryption session random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x81 | Byte Offset: 0x204 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.80 SOR_NV_PDISP_SOR_TMDS_HDCP_CN_MSB_0

HDCP CN MSB REGISTER

The CN_MSB register holds the 32 most significant bits of the upstream exchange random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x82 | Byte Offset: 0x208 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

27.8.81 SOR_NV_PDISP_SOR_TMDS_HDCP_CN_LSB_0

HDCP CN LSB REGISTER

The CN_LSB register holds the 32 least significant bits of the upstream exchange random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x83 | Byte Offset: 0x20c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

27.8.82 SOR_NV_PDISP_SOR_TMDS_HDCP_AKSV_MSB_0

HDCP AKSV MSB REGISTER

The AKSV_MSB register holds the 8 most significant bits of the transmitter's downstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x84 | Byte Offset: 0x210 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: 0 = ZERO

27.8.83 SOR_NV_PDISP_SOR_TMDS_HDCP_AKSV_LSB_0

HDCP AKSV LSB REGISTER

The AKSV_LSB register holds the 32 least significant bits of the transmitter's downstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x85 | Byte Offset: 0x214 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.84 SOR_NV_PDISP_SOR_TMDS_HDCP_BKSV_MSB_0

HDCP BKSV MSB REGISTER

The BKSV_MSB register holds the 8 most significant bits of the receiver's key selection vector (KSV), as well as the receiver's REPEATER bit. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x86 | Byte Offset: 0x218 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31	0x0	REPEATER: 0 = VALUE
7:0	0x0	VALUE: 0 = ZERO

27.8.85 SOR_NV_PDISP_SOR_TMDS_HDCP_BKSV_LSB_0

HDCP BKSV LSB REGISTER

The BKSV_LSB register holds the 32 least significant bits of the receiver's key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x87 | Byte Offset: 0x21c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

27.8.86 SOR_NV_PDISP_SOR_TMDS_HDCP_CKSV_MSB_0

HDCP CKSV MSB REGISTER

The CKSV_MSB register holds the 8 most significant bits of the software's key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

> ***Note:*** *Writing CKSV_MSB is the trigger for computing MPRIME and DKSV. It must be written last, after HDCP_CMODE, CN_MSB/LSB, and CKSV_LSB.*

Usage: normal operation

Offset: 0x88 | Byte Offset: 0x220 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: 0 = ZERO

27.8.87 SOR_NV_PDISP_SOR_TMDS_HDCP_CKSV_LSB_0

HDCP CKSV LSB REGISTER

The CKSV_LSB register holds the 32 least significant bits of the software's key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x89 | Byte Offset: 0x224 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

27.8.88 SOR_NV_PDISP_SOR_TMDS_HDCP_DKSV_MSB_0

HDCP DKSV MSB REGISTER

The DKSV_MSB register holds the 8 most significant bits of the transmitter's upstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x8a | Byte Offset: 0x228 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: 0 = ZERO

27.8.89 SOR_NV_PDISP_SOR_TMDS_HDCP_DKSV_LSB_0

HDCP DKSV LSB REGISTER

The DKSV_LSB register holds the 32 least significant bits of the transmitter's upstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x8b | Byte Offset: 0x22c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.90 SOR_NV_PDISP_SOR_TMDS_HDCP_CTRL_0

HDCP CTRL REGISTER

The CTRL register is used to facilitate synchronization between the display driver and hardware.

- RUN: Set by the driver to start or stop the downstream protocol
- CRYPT: Set by the driver to turn encryption on.

  Write ENABLED to enable, but only after RUN == YES. Once enabled, it is disabled by writing RUN=NO.
- DUAL_LINK_EN: Set by the driver to turn dual-link mode on or off
- ONEONE: ONEONE_EN enables the HDCP 1.1 features. This enables HDCP EESS signaling and Advance_Cipher.

  Write ENABLED to enable, but only after RUN == YES. Once enabled, it is disabled by writing RUN=NO. This should be enabled if your receiver supports HDCP 1.1. See Chapter 2.7 in the HDCP 1.1 specification for EESS signaling. See Chapter 2.2 in the HDCP 1.1 specification for ADVANCE_CIPHER.
- AN: Set by hardware when a valid An value has been generated
- R0: Set by hardware when Km, Ks, M0, and R0 have been calculated
- SPRIME: Set by hardware when S' has been calculated
- MPRIME: Set by hardware when M' has been calculated
- SROM_EN: Set by hardware while the HDCP SROM is in use, cleared when not in use

- SROM_ERR: Set by hardware when an error occurs while using the HDCP SROM

Usage: normal operation

Offset: 0x8c | Byte Offset: 0x230 | Read/Write: R/W | Reset: 0x0000XX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	R/W	Reset	Description
13	RO	X	SROM_ERR: 0 = NOERROR 1 = ERROR
12	RO	X	SROM_EN: 0 = DISABLED 1 = ENABLED
11	RO	X	MPRIME: 0 = INVALID 1 = VALID
10	RO	X	SPRIME: 0 = INVALID 1 = VALID
9	RO	X	R0: 0 = INVALID 1 = VALID
8	RO	X	AN: 0 = INVALID 1 = VALID
3	RW	0x0	ONEONE: 0 = DISABLED 1 = ENABLED
2	RW	0x0	DUAL_LINK_EN: 0 = DISABLED 1 = ENABLED
1	RW	0x0	CRYPT: 0 = DISABLED 1 = ENABLED
0	RW	0x0	RUN: 0 = NO 1 = YES

27.8.91 SOR_NV_PDISP_SOR_TMDS_HDCP_CMODE_0

HDCP CMODE REGISTER

The CMODE register indicates to hardware which upstream protocol calculation to perform. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Writing to the CMODE register kicks off a "Read Status" or a "Read M" operation. When performing a "Read Status" operation, use the CMODE_INDEX field to identify the port for which you wish to obtain status.

Usage: normal operation

Offset: 0x8d | Byte Offset: 0x234 | Read/Write: R/W | Reset: 0x00000011 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00010001)

Bit	Reset	Description
7:4	SOR1	INDEX: 0 = SOR0 1 = SOR1 2 = SOR2 3 = SOR3 4 = SOR4 5 = SOR5 6 = SOR6 7 = SOR7 8 = DAC0 9 = DAC1 10 = DAC2 11 = PIOR0 12 = PIOR1 13 = PIOR2 14 = PIOR3 15 = PIOR4 15 = MAX
3:0	0x1	MODE: 1 = READ_S 2 = READ_M

27.8.92 SOR_NV_PDISP_SOR_TMDS_HDCP_MPRIME_MSB_0

HDCP MPRIME MSB REGISTER

The MPRIME_MSB register holds the 32 most significant bits of the encrypted M0 value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x8e | Byte Offset: 0x238 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.93 SOR_NV_PDISP_SOR_TMDS_HDCP_MPRIME_LSB_0

HDCP MPRIME LSB REGISTER

The MPRIME_LSB register holds the 32 least significant bits of the encrypted M0 value. See the HDCP specification and the Upstream Link for HDCP specification document for more information.

Usage: normal operation

Offset: 0x8f | Byte Offset: 0x23c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.94 SOR_NV_PDISP_SOR_TMDS_HDCP_SPRIME_MSB_0

HDCP SPRIME MSB REGISTER

The SPRIME_MSB register holds the 8 most significant bits of the S' value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

STATUS_READZ indicates whether or not this chip implements the HDCP Upstream Spec's "Read Z" operation. Currently, "Read Z" is not implemented.

STATUS_CS indicates whether or not this chip implements the connection state (CS) register. If this field is set to STATUS_CS_NOT_IMPL, it means this chip will always follow the standard "Read Status" protocol when reporting status. If this field is set to STATUS_CS_IMPLMNTD, it means this chip will always follow the "Read Status With Connection State" protocol when reporting status.

STATUS_SCOPE indicates how the chip will report the status for the STATUS_UNPROTECTED field. See the STATUS_UNPROTECTED description below for more information.

STATUS_INTPNL indicates if the port you are querying (identified by the CMode index) is transmitting to an internal panel on this head. If the port you are querying is not transmitting on this head (see the ** note below), or if it is transmitting on this head but to a device other than an internal panel, then STATUS_INTPNL_INACTV will be returned.

STATUS_MAX_CMODE_IDX identifies the maximum CMode index allowed for requesting status.

Note: *Unlike the STATUS_UNPROTECTED field, the other STATUS fields always use the _HA_ registers to report status for ports on head A and the _HB_ registers to report status on head B. If you use the _HA_ versions of the CMODE and SPRIME registers to query the status of a port that is actually being driven by head B, the status will indicate that the port isn't transmitting at all (everything will come back as INACTV).*

Usage: normal operation

Offset: 0x90 | Byte Offset: 0x240 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	STATUS_READZ: 0 = NOT_IMPL 1 = IMPLMNTD
6	X	STATUS_CS: 0 = NOT_IMPL 1 = IMPLMNTD
5	X	STATUS_SCOPE: 0 = SCOPE_TWO_HEADS 1 = SCOPE_1_HEAD
4	X	STATUS_INTPNL: 0 = INACTV 1 = ACTV
3:0	X	STATUS_MAX_CMODE_IDX

27.8.95 SOR_NV_PDISP_SOR_TMDS_HDCP_SPRIME_LSB2_0

HDCP SPRIME LSB2 REGISTER

The SPRIME_LSB2 register holds the 32 next-most significant bits of the S' value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

STATUS_CMODE_IDX identifies the index of the port to which the status request was actually routed.
STATUS_UNPROTECTED indicates whether the port you queried (identified by the CMode index) is transmitting unprotected data. If STATUS_SCOPE is set to STATUS_SCOPE_1_HEAD, then make sure to use the _HA_ registers to see if any ports on head A are unprotected, and use the _HB_ registers to see if any ports on head B are unprotected. If STATUS_SCOPE is set to STATUS_SCOPE_2_HEADS, then STATUS_UNPROTECTED combines the status of both heads. In this case, both the _HA_ registers and the _HB_ registers will report the same value for the STATUS_UNPROTECTED field.

Value of STATUS_SCOPE	Meaning of STATUS_UNPROTECTED
_1_HEAD	UNPROTECTED_Y means that at least one port driven by this head is transmitting "unprotected" data. UNPROTECTED_N means that none of the ports driven by this head are transmitting "unprotected" data.

Value of STATUS_SCOPE	Meaning of STATUS_UNPROTECTED
_2_HEADS	UNPROTECTED_Y means that at least one port driven by either head is transmitting "unprotected" data. UNPROTECTED_N means that none of the ports driven by either head are transmitting "unprotected" data.

"Unprotected" data does not necessarily mean data that is not being encrypted. Data being transmitted out of a DAC is *always* considered unprotected. Data being transmitted out of a DVO is considered unprotected if it is not being encrypted. Data being transmitted by a TMDS link is also considered unprotected if it is not being encrypted,*except* for the case where the link feeds the internal panel of a laptop. The HDCP specification dictates that since the bus feeding an internal panel is essentially inaccessible to users, it can be considered "protected" even when sending clear data.

STATUS_EXTPNL is set to STATUS_EXTPNL_ACTV if the port identified by the CMode index is a digital interface (i.e., a DVO or TMDS interface), and it is transmitting on this head to something *other* than an internal panel. The STATUS_INTPNL field and the STATUS EXTPNL field will never be both set to ACTV for the same port. If a digital port is transmitting on this head, and it is physically connected to an internal panel, then the STATUS_INTPNL bit is set to ACTV and the STATUS_EXTPNL is set to INACTV; if it is transmitting on this head but it is *not* connected to an internal panel (even if it is not connected to anything at all), then STATUS_INTPNL is set to INACTV and STATUS_EXTPNL is set to ACTV.

STATUS_RPTR is set to STATUS_RPTR_ACTV if 1) the STATUS_EXTPNL field is set to ACTV, and 2) the REPEATER bit of the BKSV_MSB register is set to 0x1. This scenario should only occur after we are transmitting to an external HDCP-compliant device whose "repeater" bit is set.

STATUS_ENCRYPTING will be set to STATUS_ENCRYPTING_Y if the HDCP unit in this head is actually encrypting the data it receives. Note that if this bit is set, it means that *all* ports driven by this head whose data streams flow through the HDCP unit will be transmitting encrypted data, but any ports driven by this head whose data streams *bypass* the HDCP unit will be transmitting *clear* data. The DACs always use a data path that bypasses the HDCP unit and therefore must always be considered "unprotected".

Usage: normal operation

Offset: 0x91 | Byte Offset: 0x244 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	STATUS_CMODE_IDX
27	X	STATUS_UNPROTECTED: 0 = N 1 = Y
26	X	STATUS_EXTPNL: 0 = INACTV 1 = ACTV
25	X	STATUS_RPTR: 0 = INACTV 1 = ACTV
24	X	STATUS_ENCRYPTING: 0 = N 1 = Y
23:0	X	VALUE: 0 = ZERO

27.8.96 SOR_NV_PDISP_SOR_TMDS_HDCP_SPRIME_LSB1_0

HDCP SPRIME LSB1 REGISTER

The SPRIME_LSB1 register holds the 32 least significant bits of the S' value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x92 | Byte Offset: 0x248 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

27.8.97 SOR_NV_PDISP_SOR_TMDS_HDCP_RI_0

HDCP RI REGISTER

The RI register holds the 16-bit link integrity check value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x93 | Byte Offset: 0x24c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	VALUE: 0 = ZERO

27.8.98 SOR_NV_PDISP_SOR_TMDS_HDCP_CS_MSB_0

HDCP CS MSB REGISTER

The CS_MSB register holds the 8 most significant bits of the connection state. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x94 | Byte Offset: 0x250 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RESERVED

27.8.99 SOR_NV_PDISP_SOR_TMDS_HDCP_CS_LSB_0

HDCP CS LSB REGISTER

The CS_LSB register holds the 32 least significant bits of the connection state. See the HDCP specification, and the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 0x95 | Byte Offset: 0x254 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RESERVED

27.8.100 SOR_NV_PDISP_SOR_HDMI_AUDIO_EMU_RDATA0_0

HDMI_AUDIO_EMU_RDATA0 is read-only.

Offset: 0x97 | Byte Offset: 0x25c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RDATA

27.8.101 SOR_NV_PDISP_SOR_HDMI_AUDIO_EMU1_0

HDMI_AUDIO_EMU1 and EMU2 allow control over S/PDIF transmitter clock ratios (which are fixed in the GPU). They do indirect addressing.

EMU1 has the address, and EMU2 has the data.

To write:

1. Write data register with data
2. Write address register with bit 31=1, and address in 15:0
3. Write address register with bit 31=0

To read:

1. Write address register with bit 31=0 and address in 15:0
2. Read back address register to introduce delay allowing read data to propagate
3. Read data register

Offset: 0x98 | Byte Offset: 0x260 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31	0x0	WRITE

Bit	Reset	Description
7:0	0x0	ADDR: 0 = CNT_22 1 = CNT_24 2 = CNT_32 3 = CNT_44 4 = CNT_48 5 = CNT_88 6 = CNT_96 7 = CNT_176 8 = CNT_192 9 = JIT_05_22 10 = JIT_05_24 11 = JIT_05_32 12 = JIT_05_44 13 = JIT_05_48 14 = JIT_05_88 15 = JIT_05_96 16 = JIT_05_176 17 = JIT_05_192 18 = JIT_10_22 19 = JIT_10_24 20 = JIT_10_32 21 = JIT_10_44 22 = JIT_10_48 23 = JIT_10_88 24 = JIT_10_96 25 = JIT_10_176 26 = JIT_10_192 27 = JIT_11_22 28 = JIT_11_24 29 = JIT_11_32 30 = JIT_11_44 31 = JIT_11_48 32 = JIT_11_88 33 = JIT_11_96 34 = JIT_11_176 35 = JIT_11_192 36 = GLITCH_PERIOD_22 37 = GLITCH_PERIOD_24 38 = GLITCH_PERIOD_32 39 = GLITCH_PERIOD_44 40 = GLITCH_PERIOD_48 41 = GLITCH_PERIOD_88 42 = GLITCH_PERIOD_96 43 = GLITCH_PERIOD_176 44 = GLITCH_PERIOD_192 45 = GLITCH_CNT_22 46 = GLITCH_CNT_24 47 = GLITCH_CNT_32 48 = GLITCH_CNT_44 49 = GLITCH_CNT_48 50 = GLITCH_CNT_88 51 = GLITCH_CNT_96 52 = GLITCH_CNT_176 53 = GLITCH_CNT_192

27.8.102 SOR_NV_PDISP_SOR_HDMI_AUDIO_EMU2_0

Offset: 0x99 | Byte Offset: 0x264 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

27.8.103 SOR_NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_CTRL_0

HDMI Registers

This is the register space for the HDMI block. Refer to these sections in the HDMI Version 1.1 specification:

- 5.3, Data Island Packet Definition

- 8.2.1, Auxiliary Video Information (AVI) InfoFrames
- 8.2.2, Audio InfoFrame

Also refer to IEC60958-1 for general information and IEC60958-3 for consumer specifications for related audio information.

Additional notes regarding the ACR packet:

There are several methods that can be used to generate the N and CTS values in the Audio Clock Regeneration packet. The details of these methods are described here.

Full Software control: Software controls the contents of the possible ACR packets. Software writes the contents of the ACR_XXXX_SUBPACK_HIGH and ACR_XXXX_SUBPACK_LOW for all seven audio frequencies.

Software selects a method in ACR_CTRL to determine the audio sampling frequency

1. Measured in the audio block (MEASURE)
2. Read from the channel status information encoded in the audio stream (PACKET)
3. Defined by software (FREQS)

Software writes _YES to the ENABLE field of the SUBPACK_HIGH registers to allow that pair of registers to be used as the ACR packet. The audio sampling frequency specified in ACR_CTRL will select one of the seven pairs of SUBPACK registers to use as the ACR packet. This packet is sent on every Frame 27 of the audio block if that pair's ENABLE field is _YES.

Hardware controlled CTS: Software provides the N value and the hardware measures the CTS value. This is described in the HDMI_SPARE comments and mentioned in all other affected registers.

Software writes _ENABLE to HDMI_SPARE_HW_CTS to enable this feature.

Software writes 1 to HDMI_SPARE_CTS_RESET_VAL. This controls what the internal CTS counter resets to when it starts the next round of measurement. Resetting to zero was not correct, and adding a few bits of control helped to fine tune the measurement.

Software sets all ENABLEs in ACR_CTRL to _NO. (The easy way to do this is just to write zero to the whole register.)

Software writes the value of N in two places:

1. ACR_0441_SUBPACK_HIGH_SB[4/5/6]: It is written here so that the ACR packet contains the correct information for N.
2. AUDIO_N_VALUE: It is written here so that the audio block knows what N to use while it is measuring CTS. (Software does not need to write N to these places if Hardware Selected N is used.)

Software enables the sending of ACR_0441_SUBPACK* by setting ACR_0441_SUBPACK_ENABLE to _YES. (This pair of registers is used for all audio frequencies when Hardware controlled CTS is enabled.)

Software writes _USE_HW_CTS_VAL to ACR_0441_SUBPACK_LOW_SB1.

Hardware selected N: Software provides the possible N values whenever the pixel clock frequency changes. Hardware looks up the correct N value from a table stored in priv registers.

Software writes the 7 possible N values into the 7 AUDIO_NVAL registers. This is based on pixel clock frequency.

Software enables this feature by setting AUDIO_N_LOOKUP to _ENABLE. Hardware will use the audio sampling frequency detected by the audio block to select the correct N value.

Note: *This is not the audio sampling frequency reported by the channel status information in the audio stream. Hardware will ignore whatever if written in the SUBPACK_HIGH registers when this feature is enabled and send the value of N it selected in these bytes of the ACR packet. CTS can come from the SUBPACK_LOW registers or the Hardware controlled CTS.*

Audio frequency notes: There are several places in the HDMI and Audio registers where the audio sampling frequency is reported or set. Here is a breakdown of all such fields.

In SPDIF audio encoding, one bit of each audio sample is part of the "channel status" information. The bits from all 192 audio samples combine to form the 192 bit channel status packet. For consumer applications, only the first 40 bits have defined values. These bits are included in the encoded HDMI audio packet. Four of these bits contain the audio sampling frequency as reported by the audio source. See the IEC60958-3 specification for more information.

SPDIF_CHN_STATUS1 and SPDIF_CHN_STATUS2 are read-only registers that contain the 40 bits of channel status data read from the last audio block. The audio frequency the audio stream reports can be read from SPDIF_CHN_STATUS1_SFREQ. This is the sampling frequency that ACR_CTRL refers to with _PACKET.

SPDIF_CHN_STATUS1_ORIGINAL is the original sampling frequency of the audio. If the source of the audio stream changed the sampling frequency from its original frequency for any reason, the original sampling frequency is reported here.

AUDIO_CNTRL0_SAMPLING_FREQ is the sampling frequency measured by the audio block. It counts the number of dispclks periods that occur between to specific points in the audio stream. It compares this count to the thresholds in the AUDIO_FS registers to determine what the audio sampling frequency is. This is the sampling frequency that ACR_CTRL refers to as _MEASURE. This is the sampling frequency used to determine the correct value of N when Hardware selected N is enabled (see AUDIO_N).

The CHANNEL_STATUS1 and CHANNEL_STATUS2 registers are for debug. They can be used to replace the channel status bits that were in the original SPDIF stream with user-defined bits. CHANNEL_STATUS1_SFREQ can be used to report a different sampling frequency to downstream devices.

ACR_CTRL_FREQS can be written to select a specific pair of SUBPACK registers to send as your ACR packet. ACR_CTRL_FREQS_ENABLE must be set to _YES for this to have any effect.

The HDMI Audio InfoFrame, AVI InfoFrame, and Generic InfoFrame have nearly identical priv register interfaces. The next five registers control the Audio InfoFrame as described in Section 8.2.2 of the HDMI specification.

HDMI_AUDIO_INFOFRAME_CTRL

This register controls the frequency and generation of audio InfoFrame packets. The contents of the packet should be written into the header (HDMI_AUDIO_INFOFRAME_HEADER) and subpacket (HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW, HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH) registers.

ENABLE: Setting this field to _YES will initiate InfoFrame generation. Setting this bit to _NO will disable InfoFrame generation at the beginning of the next frame. The frequency of InfoFrame generation is controlled by OTHER and SINGLE fields.

OTHER: Setting this field to _EN while SINGLE is set to _DIS will cause InfoFrame to be transmitted to every other frame.

SINGLE: Setting this field to _EN while OTHER is set to _DIS will cause InfoFrame to be transmitted exactly once.

If OTHER and SINGLE fields are both set to _DIS, an InfoFrame will be generated every frame. Software should never set OTHER and SINGLE both to _EN.

See chapter 8.2.2 - Audio InfoFrame, in the HDMI specification for more information

CHKSUM_HW: Hardware provides a way to calculate the checksum for the InfoFrames.

_ENABLE will enable the hardware calculation to be passed to the packet

_DISABLE:

- Uses the register value defined in NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW_PB0 for the S/PDIF
- Uses the checksum provided by the Azalia codec for the Azalia audio formats.

Usage: normal operation

Offset: 0x9a | Byte Offset: 0x268 | Read/Write: R/W | Reset: 0x00000200 (0bxxxxxxxxxxxxxxxxxxxxxxxx10xxx0xxx0)

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: CHKSUM_HW is an optional feature for SW to use. Not required to be enabled. 0 = DISABLE 1 = ENABLE
8	0x0	SINGLE: 0 = DIS 1 = EN
4	0x0	OTHER: 0 = DIS 1 = EN
0	0x0	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

27.8.104 SOR_NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_STATUS_0

HDMI_AUDIO_INFOFRAME_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Usage: normal operation

Offset: 0x9b | Byte Offset: 0x26c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

27.8.105 SOR_NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_HEADER_0

HDMI_A This register should be written with the value of the HDMI Audio InfoFrame header. This header is described in table 8-4 of chapter 8.2.2 of the HDMI specification. HB0, HB1, and HB2 are defined fields of the header from the specification.

Offset: 0x9c | Byte Offset: 0x270 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

27.8.106 SOR_NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW_0

HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW

This register should be written with the lower 4 bytes of the HDMI Audio InfoFrame. PB0, PB1, PB2, and PB3 are defined fields from the specification. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI specification for more information

Offset: 0x9d | Byte Offset: 0x274 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

27.8.107 SOR_NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH_0

HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH

This register should be written with the upper 2 bytes of the HDMI Audio InfoFrame. PB4 and PB5 are defined fields from the specification. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI specification for more information

Offset: 0x9e | Byte Offset: 0x278 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PB5
7:0	0x0	PB4

27.8.108 SOR_NV_PDISP_SOR_HDMI_AVI_INFOFRAME_CTRL_0

The following registers are used to send a 14-byte packet intended for sending AVI InfoFrame packet as described in Section 8.2.1 of the HDMI specification.

HDMI_AVI_INFOFRAME_CTRL

This register controls the frequency and generation of AVI InfoFrame packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL. See Section 8.2.1 - Auxiliary Video information (AVI) InfoFrame, in the HDMI specification for more information.

Offset: 0x9f | Byte Offset: 0x27c | Read/Write: R/W | Reset: 0x00000200 (0bxxxxxxxxxxxxxxxxxxxxxxxx10xxx0xxx0)

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: CHKSUM_HW is an optional feature for software to use. Not required to be enabled. 0 = DISABLE 1 = ENABLE
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

27.8.109 SOR_NV_PDISP_SOR_HDMI_AVI_INFOFRAME_STATUS_0

HDMI_AVI_INFOFRAME_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0xa0 | Byte Offset: 0x280 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

27.8.110 SOR_NV_PDISP_SOR_HDMI_AVI_INFOFRAME_HEADER_0

HDMI_AVI_INFOFRAME_HEADER

This register should be written with the value of the HDMI AVI InfoFrame header. This header is described in table 8-1 of chapter 8.2.1 of the HDMI specification. HB0, HB1, and HB2 are defined fields of the header from the specification.

Offset: 0xa1 | Byte Offset: 0x284 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

27.8.111 SOR_NV_PDISP_SOR_HDMI_AVI_INFOFRAME_SUBPACK0_LOW_0

HDMI_AVI_INFOFRAME_SUBPACK0_LOW

This register should be written with the lower 4 bytes of the HDMI AVI InfoFrame. PB0, PB1, PB2, and PB3 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0xa2 | Byte Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

27.8.112 SOR_NV_PDISP_SOR_HDMI_AVI_INFOFRAME_SUBPACK0_HIGH_0

HDMI_AVI_INFOFRAME_SUBPACK0_HIGH

This register should be written with bytes 4-6 of the HDMI AVI InfoFrame. PB3, PB5, and PB6 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0xa3 | Byte Offset: 0x28c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

27.8.113 SOR_NV_PDISP_SOR_HDMI_AVI_INFOFRAME_SUBPACK1_LOW_0

HDMI_AVI_INFOFRAME_SUBPACK1_LOW

This register should be written with bytes 7-10 of the HDMI AVI InfoFrame. PB7, PB8, PB9, and PB10 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0xa4 | Byte Offset: 0x290 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8

Bit	Reset	Description
7:0	0x0	PB7

27.8.114 SOR_NV_PDISP_SOR_HDMI_AVI_INFOFRAME_SUBPACK1_HIGH_0

HDMI_AVI_INFOFRAME_SUBPACK1_HIGH

This register should be written with bytes 11-13 of the HDMI AVI InfoFrame. PB11, PB12, and PB13 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0xa5 | Byte Offset: 0x294 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

27.8.115 SOR_NV_PDISP_SOR_HDMI_GENERIC_STATUS_0

HDMI_GENERIC_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0xa7 | Byte Offset: 0x29c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

27.8.116 SOR_NV_PDISP_SOR_HDMI_GENERIC_HEADER_0

HDMI_GENERIC_HEADER

This register should be written with the contents of the packet header.

Offset: 0x2c | Byte Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

27.8.117 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK0_LOW_0

HDMI_GENERIC_SUBPACK0_LOW

Bytes 0-3 of the packet are written into this register.

Offset: 0xa9 | Byte Offset: 0x2a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

27.8.118 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK0_HIGH_0

HDMI_GENERIC_SUBPACK0_HIGH

Bytes 4-6 of the packet are written into this register.

Offset: 0xaa | Byte Offset: 0x2a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

27.8.119 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK1_LOW_0

HDMI_GENERIC_SUBPACK1_LOW

Bytes 7-10 of the packet are written into this register.

Offset: 0xab | Byte Offset: 0x2ac | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

27.8.120 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK1_HIGH_0

HDMI_GENERIC_SUBPACK1_HIGH

Bytes 11-13 of the packet are written into this register.

Offset: 0xac | Byte Offset: 0x2b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

27.8.121 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK2_LOW_0

HDMI_GENERIC_SUBPACK2_LOW

Bytes 14-17 of the packet are written into this register.

Offset: 0xad | Byte Offset: 0x2b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB17
23:16	0x0	PB16
15:8	0x0	PB15
7:0	0x0	PB14

27.8.122 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK2_HIGH_0

HDMI_GENERIC_SUBPACK2_HIGH

Bytes 18-20 of the packet are written into this register.

Offset: 0xae | Byte Offset: 0x2b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB20
15:8	0x0	PB19
7:0	0x0	PB18

27.8.123 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK3_LOW_0

HDMI_GENERIC_SUBPACK3_LOW

Bytes 21-24 of the packet are written into this register.

Offset: 0xaf | Byte Offset: 0x2bc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB24
23:16	0x0	PB23
15:8	0x0	PB22
7:0	0x0	PB21

27.8.124 SOR_NV_PDISP_SOR_HDMI_GENERIC_SUBPACK3_HIGH_0

HDMI_GENERIC_SUBPACK3_HIGH

Bytes 25-27 of the packet are written into this register.

Offset: 0xb0 | Byte Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB27
15:8	0x0	PB26
7:0	0x0	PB25

27.8.125 SOR_NV_PDISP_SOR_HDMI_ACR_CTRL_0

HDMI_ACR_CTRL

The Audio Clock Regeneration (ACR) packet contains the N and CTS values that the HDMI sink requires to recreate the audio clock. This mechanism is described in chapter 7.2 of the HDMI specification.

If software needs to specify N and CTS directly, this register is used to select the audio sampling frequency detection mechanism. The sampling rate is used to index one of the seven ACR_XXXX_SUBPACK_LOW/HIGH registers which must be written with the correct value of N and CTS. N and CTS are determined by the current audio sampling frequency and the pixel clock frequency:

CTS = (PixelClock*N) / (128*AudioSamplingFrequency)

The selected sampling rate must point to a valid entry (i.e., one of the 7 listed below) and the selected ACR_XXXX_SUBPACK_HIGH_ENABLE must be set to enable sending the packet.

When software is controlling N and CTS directly, the ACR packet is sent every 27th audio frame (of 0 to 191) of the audio block. The sampling frequency read from the channel status bits is not available until the 27th audio frame.

PACKET_ENABLE: Set this to _YES to use the channel status information read from the incoming SPDIF audio stream to determine the sampling frequency.

MEASURE_ENABLE: Set this to _YES to use the sampling frequency measured in the in the audio block to determine the sampling frequency. This is the sampling frequency that is read from AUDIO_CNTRL0_SAMPLING_FREQ.

FREQS_ENABLE: Set this to _YES to use the sampling frequency written into the _FREQS field of this register.

FREQS: This is the audio sampling frequency that will be used when FREQS_ENABLE is _YES.

When Hardware is used to measure CTS, all ENABLE fields of this register should be set to _NO.

All four subpackets contain the same ACR packet.

For more information, see the following in the HDMI specification:

- 7.2: Audio Sample Clock Capture and Regeneration use the other entry in tables 7-1, 7-2 and 7-3 to determine N
- 5.3.3: Audio Clock Regeneration Packet

Offset: 0xb1 | Byte Offset: 0x2c4 | Read/Write: R/W | Reset: 0x02010000 (0bxxxx0010xxxxxxx1xxxxxxx0xxxxxxx0)

Bit	Reset	Description
27:24	FREQ_48KHZ	FREQS: 3 = FREQ_32KHZ 0 = FREQ_44_1KHZ 2 = FREQ_48KHZ 8 = FREQ_88_2KHZ 10 = FREQ_96KHZ 12 = FREQ_176_4KHZ 14 = FREQ_192KHZ
16	YES	FREQS_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
8	NO	MEASURE_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
0	NO	PACKET_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

27.8.126 SOR_NV_PDISP_SOR_HDMI_ACR_0320_SUBPACK_LOW_0

HDMI_ACR_0320_SUBPACK_LOW

Contains bytes 1-3 of the 32 kHz ACR packet. This is the CTS value.

See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0xb2 | Byte Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

27.8.127 SOR_NV_PDISP_SOR_HDMI_ACR_0320_SUBPACK_HIGH_0

HDMI_ACR_0320_SUBPACK_HIGH

Contains bytes 4-6 of the 32 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0xb3 | Byte Offset: 0x2cc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

27.8.128 SOR_NV_PDISP_SOR_HDMI_ACR_0441_SUBPACK_LOW_0

HDMI_ACR_0441_SUBPACK_LOW

Contains bytes 1-3 of the 44.1 kHz ACR packet. This is the CTS value. If Hardware measured CTS is enabled, SB1 should be set to zero. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0xb4 | Byte Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1: 0 = USE_HW_CTS_VAL
23:16	0x0	SB2
15:8	0x0	SB3

27.8.129 SOR_NV_PDISP_SOR_HDMI_ACR_0441_SUBPACK_HIGH_0

HDMI_ACR_0441_SUBPACK_HIGH

Contains bytes 4-6 of the 44.1 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

If Hardware measured CTS is enabled, ACR_0441_SUBPACK_HIGH_N should be written with the N value for the current audio sampling frequency.

If the hardware N value selection is enabled, the N value does not need to be written to this register.

ENABLE: _YES allows this packet to be sent. This should be set to _YES when hardware measured CTS is being used. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0xb5 | Byte Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

27.8.130 SOR_NV_PDISP_SOR_HDMI_ACR_0882_SUBPACK_LOW_0

HDMI_ACR_0882_SUBPACK_LOW

Contains bytes 1-3 of the 88.2 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0xb6 | Byte Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

27.8.131 SOR_NV_PDISP_SOR_HDMI_ACR_0882_SUBPACK_HIGH_0

HDMI_ACR_0882_SUBPACK_HIGH

Contains bytes 4-6 of the 88.2 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0xb7 | Byte Offset: 0x2dc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

27.8.132 SOR_NV_PDISP_SOR_HDMI_ACR_1764_SUBPACK_LOW_0

HDMI_ACR_1764_SUBPACK_LOW

Contains bytes 1-3 of the 176.4 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0xb8 | Byte Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

27.8.133 SOR_NV_PDISP_SOR_HDMI_ACR_1764_SUBPACK_HIGH_0

HDMI_ACR_1764_SUBPACK_HIGH

Contains bytes 4-6 of the 176.4 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0xb9 | Byte Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

Bit	Reset	Description
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

27.8.134 SOR_NV_PDISP_SOR_HDMI_ACR_0480_SUBPACK_LOW_0

HDMI_ACR_0480_SUBPACK_LOW

Contains bytes 1-3 of the 48 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0xba | Byte Offset: 0x2e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

27.8.135 SOR_NV_PDISP_SOR_HDMI_ACR_0480_SUBPACK_HIGH_0

HDMI_ACR_0480_SUBPACK_HIGH

Contains bytes 4-6 of the 48 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0xbb | Byte Offset: 0x2ec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

27.8.136 SOR_NV_PDISP_SOR_HDMI_ACR_0960_SUBPACK_LOW_0

HDMI_ACR_0960_SUBPACK_LOW

Contains bytes 1-3 of the 96 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 of the HDMI specification for more information.

Offset: 0xbc | Byte Offset: 0x2f0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

27.8.137 SOR_NV_PDISP_SOR_HDMI_ACR_0960_SUBPACK_HIGH_0

HDMI_ACR_0960_SUBPACK_HIGH

Contains bytes 4-6 of the 96 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0xbd | Byte Offset: 0x2f4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

27.8.138 SOR_NV_PDISP_SOR_HDMI_ACR_1920_SUBPACK_LOW_0

HDMI_ACR_1920_SUBPACK_LOW

Contains bytes 1-3 of the 192 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0xbe | Byte Offset: 0x2f8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

27.8.139 SOR_NV_PDISP_SOR_HDMI_ACR_1920_SUBPACK_HIGH_0

HDMI_ACR_1920_SUBPACK_HIGH

Contains bytes 4-6 of the 192 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0xbf | Byte Offset: 0x2fc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

27.8.140 SOR_NV_PDISP_SOR_HDMI_CTRL_0

HDMI_CTRL

REKEY: REKEY is the number of clocks required for HDCP rekey, starting from when DE is deasserted. No HDMI packets can be sent during this time. Due to a two cycle delay in hardware, REKEY should be set to two less than the desired value.

AUDIO_LAYOUT: AUDIO_LAYOUT controls layout (HB1[4]) of the Audio Sample Packet Header. Two different layouts are supported in the HDMI specification. The hardware only supports layout _2CH (2 channel audio). This should not need to be programmed beyond its init value of _2CH.

See Section 5.3.4 of the HDMI specification.

AUDIO_LAYOUT_SELECT: For projects with integrated audio codec, the AUDIO_LAYOUT information is automatically detected by hardware (default). We can override this capability and fall back on AUDIO_LAYOUT based selection by setting AUDIO_LAYOUT_SELECT to _SW_BASED.

If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in the project specification, this field has no meaning, and only AUDIO_LAYOUT field will take effect.

SAMPLE_FLAT: SAMPLE_FLAT controls the values of (HB2[3:0]) of the Audio Sample Packet Header and should be _CLR. See Section 5.3.4 of the HDMI specification.

MAX_AC_PACKET: Set MAX_AC_PACKET to the maximum number of 32-pixel packets that will fit in the horizontal blanking interval. This controls the maximum number of audio packets, ACR packets, GCP packets, InfoFrames, etc. that will be sent during the horizontal blanking period.

MAX_AC_PACKET <= Floor[((HBLANK-REKEY-18)/32)]

CT_SELECT: Diagnostic workaround bit to decide whether the value of "coding type" will be hardware based or software based. The CT field is required in the construction of Audio InfoFrame packet (HDMI Specification, Section 8.2.2). By default, the CT field is constructed by software. However, by setting this bit, the CT field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

CC_SELECT: Diagnostic workaround bit to decide whether the value of "channel count" will be hardware based or software based. The CC field is required in the construction of Audio InfoFrame packet (HDMI Specification, Section 8.2.2). By default, the CC field is constructed by software. However,by setting this bit, the CC field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

SF_SELECT: Diagnostic workaround bit to decide whether the value of "sampling frequency" will be hardware based or software based. The SF field is required in the construction of Audio InfoFrame packet (HDMI Specification, Section 8.2.2). By default, the SF field is constructed by software. However, by setting this bit, the SF field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

SS_SELECT: Diagnostic workaround bit to decide whether the value of "sample size" will be hardware based or software based. The SS field is required in the construction of Audio InfoFrame packet (HDMI Specification, Section 8.2.2). By default, the SS field is constructed by software. However, by setting this bit, the SS field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

CA_SELECT: Diagnostic workaround bit to decide whether the value of "channel allocation" will be hardware based or software based. The CA field is required in the construction of Audio InfoFrame packet (HDMI Specification, Section 8.2.2). By default, the CA field is constructed by software. However, by setting this bit, the CA field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

ENABLE: Set to _YES to enable HDMI for this head. Set to _NO to disable HDMI for this head.

Offset: 0xc0 | Byte Offset: 0x300 | Read/Write: R/W | Reset: 0x00020038 (0bx0x00000xxx00010xxx0x0x0x0111000)

Bit	Reset	Description
30	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
28	SW	CA_SELECT: 0 = SW 1 = HW
27	SW	SS_SELECT: 0 = SW 1 = HW
26	SW	SF_SELECT: 0 = SW 1 = HW
25	SW	CC_SELECT: 0 = SW 1 = HW
24	SW	CT_SELECT: 0 = SW 1 = HW
20:16	0x2	MAX_AC_PACKET
12	CLR	SAMPLE_FLAT: 0 = CLR 1 = SET
10	HW_BASED	AUDIO_LAYOUT_SELECT: 0 = HW_BASED 1 = SW_BASED
8	LAYOUT_2CH	AUDIO_LAYOUT: 0 = LAYOUT_2CH 1 = LAYOUT_8CH
6:0	0x38	REKEY

27.8.141 SOR_NV_PDISP_SOR_HDMI_VSYNC_KEEPOUT_0

HDMI_VSYNC_KEEPOUT

Defines the start and end of the VSYNC keepout period where HDMI packets should not be sent. This is defined in chapter 2.7 the HDCP 1.1 specification.

END: Defines the end of the keepout period. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

START: Defines the start of the keepout period. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

ENABLE: When set to _YES, the keepout window is respected and no HDMI packets are sent in the period of time between START and END. This bit should be set to _YES. When set to _NO, the keepout window is ignored and HDMI packets may be sent regardless of the keepout window.

Offset: 0xc1 | Byte Offset: 0x304 | Read/Write: R/W | Reset: 0x819a028a (0b1xxxxx0110011010xxxxxx1010001010)

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x19a	START
9:0	0x28a	END

27.8.142 SOR_NV_PDISP_SOR_HDMI_VSYNC_WINDOW_0

HDMI_VSYNC_WINDOW

Defines the start and end of the window of opportunity where the HDCP EESS signaling occurs. This is defined in chapter 2.7 of the HDCP 1.1 specification.

END: Defines the end of the window of opportunity. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

START: Defines the start of the window of opportunity. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

ENABLE: _YES will allow EESS signaling during the window of opportunity. _NO will prevent EESS signaling.

Offset: 0xc2 | Byte Offset: 0x308 | Read/Write: R/W | Reset: 0x82000210 (0b1xxxxx1000000000xxxxxx1000010000)

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x200	START
9:0	0x210	END

27.8.143 SOR_NV_PDISP_SOR_HDMI_GCP_CTRL_0

The following registers are used to send a single-byte packet intended for sending the general control packet as described in Section 5.3.6 of the HDMI specification. This control packet is used to control the AVMUTE flag.

HDMI_GCP_CTRL

This register controls the frequency and generation of GCP packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL.

All four subpackets contain the same GCP packet.

Offset: 0xc3 | Byte Offset: 0x30c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxx0xxx0)

Bit	Reset	Description
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

27.8.144 SOR_NV_PDISP_SOR_HDMI_GCP_STATUS_0

HDMI_GCP_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

START_PP indicates the Pixel phase for the start fragment number of the ACTIVEVIDEO/VSYNC/HSYNC in case of the deep color:

- ACTIVE_START_PP: The first active pixel is packed in to this Phase
(e.g., if this value is 0 then 10P0,12P0,16P0)
- HSYNC_START_PP: The first pixel following the +ve transition of the VSYNC is packed in to this Phase (e.g. if this value is 0 then 10P0,12P0,16P0)
- VSYNC_START_PP: The first pixel following the +ve transition of the HSYNC is packed in to this Phase (e.g. if this value is 0 then 10P0,12P0,16P0)

END_PP indicates the Pixel phase for the end fragment number of the ACTIVEVIDEO/VSYNC/HSYNC in case of the deep color:

- ACTIVE_END_PP: The last active pixel is packed in to this Phase (e.g. if this value is 1 then 10P1,12P1,16P1)
- HSYNC_END_PP: The first pixel following the -ve transition of the VSYNC is packed in to this Phase (e.g. if this value is 1 then 10P1,12P1,16P1)
- VSYNC_END_PP: The first pixel following the -ve transition of the HSYNC is packed in to this Phase (e.g. if this value is 1 then 10P1,12P1,16P1)

Software can use above fields to set the DEFAULT PP FIELD or PP field of the GCP Packet. For more information, refer to the HDMI 1.3 specification, Section 6.5.3.

PP3 PP2 PP1 PP0 Pixel packing Phase

==========

0 0 0 0 Phase 4 (10P4)

0 0 0 1 Phase 1 (10P1, 12P1, 16P1)

0 0 1 0 Phase 2 (10P2, 12P2)

0 0 1 1 Phase 3 (10P3)

0 1 0 0 Reserved

All other values Reserved

To set the DEFAULT phase in the GCP packet, SW can use above values to make sure RASTER is setup right.

Default_Phase field of GCP SB2:

When Default_Phase is 0, the PP bits may vary and the first pixel of each Video Data Period may vary.

When Default_Phase is 1, the following will be true:

-- The first pixel of each Video Data Period shall always have a pixel packing phase of 0 (10P0, 12P0, 16P0).

Will be pointed by ACTIVE_START_PP field above

-- The first pixel following each Video Data Period shall have a pixel packing phase of 0 (10C0, 12C0, 16C0).

Will be pointed by ACTIVE_END_PP field above

-- The PP bits shall be constant for all GCPs and will be equal to the last packing phase (10P4, 12P2, 16P1).

Confirmed by HW.

-- The first pixel following every transition of HSYNC or VSYNC shall have a pixel packing phase of 0 (10C0, 12C0, 16C0).

Will be pointed by HSYNC_START_PP field above

Will be pointed by HSYNC_END_PP field above

Will be pointed by VSYNC_START_PP field above

Will be pointed by VSYNC_END_PP field above

Offset: 0xc4 | Byte Offset: 0x310 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26:24	X	HSYNC_END_PP: 1 = PP_1 2 = PP_2 3 = PP_3 4 = PP_0
22:20	X	HSYNC_START_PP: 1 = PP_1 2 = PP_2 3 = PP_3 4 = PP_0
18:16	X	VSYNC_END_PP: 1 = PP_1 2 = PP_2 3 = PP_3 4 = PP_0
14:12	X	VSYNC_START_PP: 1 = PP_1 2 = PP_2 3 = PP_3 4 = PP_0
10:8	X	ACTIVE_END_PP: 1 = PP_1 2 = PP_2 3 = PP_3 4 = PP_0
6:4	X	ACTIVE_START_PP: 1 = PP_1 2 = PP_2 3 = PP_3 4 = PP_0
0	X	SENT: 0 = WAITING 1 = DONE

27.8.145 SOR_NV_PDISP_SOR_HDMI_GCP_SUBPACK_0

HDMI_GCP_SUBPACK

This register should be written with the contents of the general control packet.

See chapter 5.3.6 of the HDMI specification for more information.

Offset: 0xc5 | Byte Offset: 0x314 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxx00000000000000000000001)

Bit	Reset	Description
23:16	0x0	SB2
15:8	0x0	SB1
7:0	0x1	SB0: 1 = SET_AVMUTE 16 = CLR_AVMUTE

27.8.146 SOR_NV_PDISP_SOR_HDMI_EMU0_0

HDMI_EMU0 and 1 do indirect addressing. EMU0 has the address, and EMU1 has the data.

To write:

1. Write the data register with data.
2. Write the address register with bit 31=1, and the address in bits 15:0.
3. Write the address register with bit 31=0.

To read:

1. Write the address register with bit 31=0 and the address in bits 15:0.
2. Read back the address register to introduce delay allowing the read data to propagate.
3. Read the data register.

Offset: 0xc8 | Byte Offset: 0x320 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	regx

27.8.147 SOR_NV_PDISP_SOR_HDMI_EMU1_0

Offset: 0xc9 | Byte Offset: 0x324 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	regx

27.8.148 SOR_NV_PDISP_SOR_HDMI_EMU1_RDATA_0

Offset: 0xca | Byte Offset: 0x328 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	rdata

27.8.149 SOR_NV_PDISP_SOR_HDMI_SPARE_0

HDMI_SPARE

HW_CTS - If this is set to _ENABLE, Hardware measured CTS will be enabled. This method is more accurate than using software controlled CTS. The HDMI block will count the number of TMDS clocks that occur during the interval 1/(128 * audio sample rate) and use the value as CTS and issue the ACR Packet at each interval. When hardware measured CTS is enabled, all ENABLE fields of ACR_CTRL must be set to _NO. Also, ACR_0441_SUBPACK_HIGH_ENABLE must be set to _YES.

ACR_0441_SUBPACK_HIGH_N should be written with the value of N for the current sampling frequency.

The N value must also be written into AUDIO_N_VALUE (See the AUDIO register section for more details).

The value of N can be determined from the current audio sampling frequency, the current pixel clock rate, and tables 7-1. 7-2, and 7-3 in the HDMI specification. In most cases the value in the "Other" row can be used.

FORCE_SW_CTS: When HW_CTS = ENABLE, it uses measured CTS. However, when the audio clock / video clock ratio is known, this is undesirable. Currently, HW_CTS cannot be disabled.

When FORCE_SW_CTS=ENABLE along with HW_CTS=ENABLE, the HW_CTS is used to control transmission of ACR packets, but the CTS is programmed by software in the ACR_0441_SUBPACK_LOW registers.

SUPRESS_SP_B: Default SUPRESS_SP_B=0 is to disable B bits in audio sample packets, for non-present samples. Setting SUPRESS_SP_B=1 restores the old behavior.

CTS_RESET_VAL: When an ACR packet is sent, the CTS counter is reset to the value in this field. This should be set to the INIT value of 1

ACR_PRIORITY: This controls the priority of the ACR packet with respect to Audio Sample packets. This should be set to _HIGH.

LOW: ACR packets will have lower priority than Audio Sample packets

HIGH: ACR packets will have higher priority than Audio Sample packets

Software needs to make sure these registers are set when audio is active:

- HDMI_SPARE_HW_CTS = _HW_CTS_ENABLE
- HDMI_SPARE_CTS_RESET_VAL = 1
- HDMI_SPARE_ACR_PRIORITY = _HIGH
- ACR_CTRL_*_ENABLE = _NO
- ACR_0441_SUBPACK_HIGH_ENABLE = _YES
- ACR_0441_SUBPACK_HIGH_N = N
- AUDIO_N_VALUE = N
- ACR_0441_SUBPACK_LOW_SB1 = _USE_HW_CTS_VAL

Offset: 0xcb | Byte Offset: 0x32c | Read/Write: R/W | Reset: 0x00010000 (0b00000000000000010000000000000000)

Bit	Reset	Description
31:0	0x10000	

27.8.150 SOR_NV_PDISP_SOR_HDMI_SPDIF_CHN_STATUS1_0

HDMI_SPDIF_CHN_STATUS1

HDMI_SPDIF_CHN_STATUS1/2 contains the value of the channel status bits extracted from the incoming S/PDIF audio data stream. See IEC60958-3 chapter 5 for more information on these fields.

USE Specifies consumer use (CONSUMER) or professional (PRO) use of the channel status block. SPDIF_CHN_STATUS1/2 assumes consumer use.

TYPE: Specifies the type of data the audio word represents.

- PCM: Audio sample word represents linear PCM samples
- OTHER: Audio sample word is something other than liner PCM (i.e., compressed audio)

COPYRIGHT: Copyright status of the audio.

- YES: Copyright is asserted
- NO: Copyright is not asserted

D: The values of these bits have different meanings based on the value of _TYPE.

If _TYPE==_PCM, then:

- NO_PREEMPHASIS: 2 audio channels without pre-emphasis
- PREEMPHASIS: 2 audio channels with 50 microseconds/15 microseconds pre-emphasis

All other states are reserved for future use.

MODE Defines one of four possible channel status formats for bytes 1-23 of channel status. Currently only the value "0" is defined. This format is assumed for the SPDIF_CHN_STATUS1/2.

CODE: This byte defines the Category Code of the input device. This code indicates what type of equipment is generating the input. See 5.3.1 of the IEC60958-3 specification for the values of codes.

SOURCE: Source number of the audio. If this value is zero then no source number was reported.

CHANNEL: Channel number of the audio. This reports the channel number reported for the first subframe of audio. If this value is zero then the channel number was not reported.

SFREQ: This is the reported sampling frequency of the input audio stream. The value UNDEFINED means that the sampling frequency was not indicated by the audio stream.

ACCURACY: Transmitter Clock accuracy.

- LVL1 High accuracy mode: Transmitted sampling frequency is within a tolerance of +/-50*10^-6
- LVL2 Normal accuracy mode: Transmitted sampling frequency is within a tolerance of +/-1000*10^-6
- LVL3 Variable pitch shifted clock mode: Only specially designed receivers can receive the signal in this mode
- OTHER: Interface frame rate not matched to sampling frequency: This is for all other modes that do not have an embedded sampling frequency clock.

Offset: 0xcc | Byte Offset: 0x330 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	ACCURACY: 1 = LVL1 0 = LVL2 2 = LVL3 3 = OTHER
27:24	X	SFREQ: 1 = UNDEFINED
23:20	X	CHANNEL: 0 = UNDEFINED
19:16	X	SOURCE: 0 = UNDEFINED
15:8	X	CODE
7:6	X	MODE
5:3	X	D: 0 = NO_PREEMPHASIS 1 = PREEMPHASIS
2	X	COPYRIGHT: 0 = YES 1 = NO
1	X	TYPE: 0 = PCM 1 = OTHER
0	X	USE: 0 = CONSUMER 1 = PRO

27.8.151 SOR_NV_PDISP_SOR_HDMI_SPDIF_CHN_STATUS2_0

HDMI_SPDIF_CHN_STATUS2

MAX_LENGTH: Reports if the maximum audio sample word length is 20 bits or 24 bits.

LENGTH: Reports the audio sample word length of this block. The value of these bits is dependent on the value of MAX_LENGTH

- If MAX_LENGTH=_20 then refer to the MAX20_* defines.
- If MAX_LENGTH=_24 then refer to the MAX24_* defines.

All other bit combinations are reserved.

ORIGINAL: The original sampling frequency of the audio data. UNDEFINED indicates that the original sampling frequency was not defined

Offset: 0xcd | Byte Offset: 0x334 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:4	X	ORIGINAL: 0 = UNDEFINED

Bit	Reset	Description
3:1	X	LENGTH: 0 = MAX20_UNDEF 1 = MAX20_16BITS 2 = MAX20_18BITS 4 = MAX20_19BITS 5 = MAX20_20BITS 6 = MAX20_17BITS 0 = MAX24_UNDEF 1 = MAX24_20BITS 2 = MAX24_22BITS 4 = MAX24_23BITS 5 = MAX24_24BITS 6 = MAX24_21BITS
0	X	MAX_LENGTH

27.8.152 SOR_NV_PDISP_HDCPRIF_ROM_CTRL_0

HDCPRIF REGISTERS

This is the register space for the HDCP ROM interface control registers. These register control writing the contents of the HDCPRIF_KEY registers (56 bits) into the local key store. The keys are decoded by the CD, which outputs them on the internal key data bus. The key registers are then written by overloading the priv reg bus with the key data when a write to either key register is executed. The key data must then be input into the key store using the control commands below.As no access to key data from public buses is allowed, all these operations are performed as hardware controlled functions that are initiated by software.

Typical operation is as follows:

1. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_DISABLED, WRITE5_INIT, WRITE7_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO.
2. Decode the downstream KSV key data in the CD block.
3. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes. This operation will be overloaded with the key data from the CD block.
4. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
5. Decode the downstream key in the CD block.
6. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes. This operation will be overloaded with the key data from the CD block.
7. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE5_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
8. Repeat 5 through 7 39 more times. There are always 40 keys and one KSV value. If and only if upstream key authentication is required do the following:
9. Decode the upstream KSV key data in the CD block.
10. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes. This operation will be overloaded with the key data from the CD block.
11. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED,

WRITE5_TRIGGER, WRITE7_INIT, WRITE1_INIT, LOAD_SELECT_ZERO,

LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.

12. Decode the first upstream key in the CD block.
13. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes.

    This operation will be overloaded with the key data from the CD block.
14. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE5_INIT, WRITE1_INIT, LOAD_SELECT_ZERO,

    LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
15. Repeat 12 through 14 39 more times. There are always 40 keys and one KSV value.

This leaves the store as follows (the required format):

- Byte 0 to Byte 4: downstream KSV
- Byte 5 to Byte 11: 1'st downstream key
- Byte 12 to Byte 18: 2'nd downstream key

  ...
- Byte 278 to Byte 284: 40'th downstream key
- Byte 285 to Byte 289: upstream KSV
- Byte 290 to Byte 296: 1'st upstream key

  ...
- Byte 563 to Byte 569: 40'th upstream key

HDCPRIF_ROM_CTRL

This register controls the number of Ack Attempts that the Crypto-ROM interface will try before timing out a Command transaction on the interface. This value is used to meet the tWr requirement of the Crypto-ROM, which limits the minimum time between these transactions. tWr is the minimum time between writes, and for Crypto-ROM operations the Commands are allowed to be issued two deep, so we need to wait 2 * tWr. The interface state machine will wait for a time of (Xp/2 * 949 * ACK_ATTEMPTS), where Xp is the Xtal clock period, and 949 is the number of Xtal clock edges the RTL needs to see to complete an Command/Acknowledge Poll sequence. The _INIT value will give a timeout of approximately 18 ms with a 27 MHz xtal (which will allow us to cope with Atmel parts that are out of spec on tWr).

Offset: 0xcf | Byte Offset: 0x33c | Read/Write: R/W | Reset: 0x000003ff (0bxxxxxxxxxxxxxxxx0000001111111111)

Bit	Reset	Description
15:0	0x3ff	ACK_ATTEMPTS

- AUDIO_CNTRL0
- ERROR_TOLERANCE: Because the (dispclk_audio_fs / audio sampling frequency) is a non-integer in general and due to the spdif input stream's jitter, THREE_HALF is not exactly equal to 3 times of HALF. ERROR_TOLERANCE is the error count in dispclk periods allowed for 3 x HALF vs THREE_HALF. Based on simulation, the value of 4 to 7 is a good value when dispclk_audio_fs = 416 MHz.

  It is anticipated that software doesn't need to change this field from its init value.
- SAMPLING_FREQ: This reports the incoming spdif audio stream sampling frequency. The HDMI specification only supports 7 audio sample frequency (fs); anything other than those 7 fs will be reported as UNKNOWN.

  See HDMI 1.1 specification Table 7-4 (page 77)
- SOURCE_SELECT: Determines whether to use the SPDIF or Azalia (HDAL) audio input.

This field has no effect on chips without the HDAL input. The default is AUTO: selects SPDIF audio until the HDAL audio input is initialized by the external controller

- FRAMES_PER_BLOCK: FRAMES_PER_BLOCK is number of frames per block. By specification each block has 192 (0xC0) frames.
- AUDIO_N
- N_VALUE: N_VALUE is the N parameter in HDMI Audio Clock Regeneration Packet. This value should be written when Hardware measured CTS is enabled. The correct N value can be read from tables 7-1, 7-2, and 7-3 in the HDMI specification.
- N_RESET: N_RESET is reset for N counter, If software is controlling the value of N, every time the audio stream changes sampling freq (fs), software (RM) need to reset the N counter by writing _ASSERT to N_RESET and follow by writing a _DEASSERT to N_RESET.

  If the Hardware selected N feature is enabled (N_LOOKUP = _ENABLE), Software only needs to reset the N counter when writing to the AUDIO_NVAL registers and when enabling/disabling N_LOOKUP.

  a. set N_RESET = _ASSERT

  b. modify N value registers

  c. set N_RESET = _DEASSERT

  Modifying N_VALUE can be done in step 1.
- N_LOOKUP

  When set to _ENABLE, the hardware will select the appropriate value of N to use from one of the AUDIO_NVAL registers. This selection is based on the audio sampling frequency detected by the audio block. This is not the sampling frequency reported in the channel status bits. N_RESET should be toggled when this feature is enabled.

  When set to _DISABLE, software must program the correct N value into AUDIO_N.

27.8.153 SOR_NV_PDISP_HDCPRIF_ROM_TIMING_0

This register controls the timing of the cryptorom interface. Any time the hdmi_clk period changes, the values here must be changed too. The cryptorom bit period must more than 1 uS. 500 KHz is a safe target. The hdmi_clk frequency is reduced by the scale factor defined by the PRESCALE divider; the period is equal to (PRESCALE+1) hdmi_clk ticks.

BIT_PERIOD defines the timing of the SCL (serial clock) output. The SCL pin will be driven high for counts 0.. (BIT_PERIOD >> 1), and will be driven low for counts (BIT_PERIOD >> 1)+1 .. BIT_PERIOD.

DATA_DLY defines how many scaled ticks pass before SDA (the data bit) makes a transition after the falling edge of SCL (for data writes and for ACK'ing after data reads).

START_DLY defines how many scaled ticks pass before SDA (the data bit) makes a transition after the rising edge of SCL (for START and STOP).

The defaults make sure even with an HDMI clock of 150 MHz that the bit clock will not be faster than 500 KHz.

In general, programming should be something like this. Given the hdmi_clk frequency and the target I2C bit rate, figure out the necessary prescale. The prescale must be large enough such that the BIT_PERIOD divisor fits into 8 bits, but it should otherwise be as small as possible in order to minimize the time quantization.

(PRESCALE+1)*(BIT_PERIOD+1)*(hdmi_clk period) >= (I2C bit period)

BIT_PERIOD+1 can be as large as 256. Rearranging,

PRESCALE = ceil[(I2C bit period)/(hdmi_clk period) / 256] - 1

= ceil[(hdmi_clk frequency)/(I2C frequency) / 256] - 1

This determines the time quantum of the other divisors.

prescale_period = (hdmi_clk period) / (PRESCALE + 1)

Next, determine the bit period:

BIT_PERIOD = ceil[(hdmi_clk frequency) / (PRESCALE+1) / (I2C frequency)] - 1

SDA (data) normally changes only while SCL is low. The Atmel datasheet says that SDA must hold for at least 10ns after SCL falls and must be set up at least 100 ns before SCL rises. Thus, DATA_DLY must be:

DATA_DLY > floor(BIT_PERIOD/2) + 10 ns

DATA_DLY < BIT_PERIOD - 100 ns

In most circumstances setting DATA_DLY to arrive 75% of the way through the cycle is OK.

DATA_DLY = floor((BIT_PERIOD+1) * 3 / 4) - 1

The start and stop conditions extend the SCL high time before the next falling SCL transition. One Atmel datasheet says that SDA (data) must not change within 200 ns of the rising or falling edge of SCL. A safe calculation is to assume that we wait a tick less than half a clock period:

START_DLY = floor(BIT_PERIOD / 2) - 2

Offset: 0xe5 | Byte Offset: 0x394 | Read/Write: R/W | Reset: 0x0f080e13 (0bxxxx11110000100000011000010011)

Bit	Reset	Description
27:24	0xf	PRESCALE
23:16	0x8	START_DLY: used for start and stop timing
15:8	0xe	DATA_DLY
7:0	0x13	BIT_PERIOD

27.8.154 SOR_NV_PDISP_SOR_REFCLK_0

SOR_REFCLK

The HDMI clock is programmable and varies, depending on screen resolution. The NV_PDISP_SOR_SEQ_INSTn instructions (above) can wait for certain time intervals to elapse. Whenever hdmi_clk frequency is changed, this register must be reprogrammed to divide hdmi_clk to produce a 1 µs time reference, otherwise the time intervals requested by NV_PDISP_SOR_SEQ_INSTn will not be accurate.

The format of DIVISOR is an unsigned 8.2 divider. Because this is a simple digital divider, not a PLL, fractional values result in a jitter of one hdmi_clk between successive "1 µs" intervals, but the long-term average works out to the requested divisor. This jitter is OK because the NV_PDISP_SOR_SEQ_INST wait intervals do not need to be exact. If the integer part is written as 0, it will be interpreted the same as "1".

Offset: 0xe6 | Byte Offset: 0x398 | Read/Write: R/W | Reset: 0x00001900 (0bxxxxxxxxxxxxxxxxx0001100100xxxxxx)

Bit	Reset	Description
15:8	0x19	DIV_INT: default: 27 MHz
7:6	0x0	DIV_FRAC

27.8.155 SOR_NV_PDISP_CRC_CONTROL_0

CRC_CONTROL

The main CRC enable bit flows along the pipeline. This register controls the injection of this bit at the top of the pipeline. This register is double-buffered. The active value is updated at start of each frame from this 'ARM' register.

Other registers such as SOR_STATE2.ASY_CRCMODE and SOR_*CRC* control the actual CRC logic, once enabled.

ARM_CRC_CONTROL enables or disables computation of CRC, effective at the start of next frame.

Offset: 0xe7 | Byte Offset: 0x39c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	NO	ARM_CRC_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

27.8.156 SOR_NV_PDISP_INPUT_CONTROL_0

INPUT_CONTROL

HDMI_SRC_SELECT selects from which of the two display units to take input. This register should be changed only when HDMI is idle.

ARM_RGB_RANGE controls whether R/G/B values of 0 and 255 are permitted (FULL), or removed by clamping to [1,254] (LIMITED).

According to the EIA/CEA-861-B and HDMI specifications, the 640x480 VGA mode uses FULL, and all others use LIMITED.

Note that this does not scale the video or change the black and white points (VGA is [0,255], others are [16,235]). That must be done, if necessary, in display or at the source.

See the HDMI 1.2a specification, Section 6.6.

This register is double-buffered and will take effect on next frame boundary.

Offset: 0xe8 | Byte Offset: 0x3a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001

Bit	Reset	SW Default	Description
1	FULL	_NONE_	ARM_VIDEO_RANGE: 0 = FULL 1 = LIMITED
0	DISPLAY	DISPLAYB	HDMI_SRC_SELECT: 0 = DISPLAY 1 = DISPLAYB

27.8.157 SOR_NV_PDISP_SCRATCH_0

SCRATCH

This register is not used by hardware. It is available for software to store the state or for testing purposes.

Offset: 0xe9 | Byte Offset: 0x3a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

27.8.158 SOR_NV_PDISP_KEY_CTRL_0

HDCP KEY SRAM Register Control

This is the register space for the HDCP ROM interface control register. This register controls writing the contents of the hdcp_key_data bus (56 bits) into the local key store. The keys are decoded in the Crypto block, and then the key data is presented on the bus.

LOCAL: If this bit is ENABLED the on-chip HDCP key store will be used by the HDCP encryption block. If DISABLED, external Crypto-ROM will be used.

AUTOINC: If this bit is ENABLED, the address written to by the WRITE16 function will auto-increment after the operation is complete. This is the normal operating mode.

WRITE16: If this bit is set to TRIGGER, the HDCP keys module will write all 16 bytes of data from the hdcp_key_data bus (sourced by the CD module) into the local key store. The bytes are written into the store at contiguous bytes pointed to by the ADDRESS field in auto-increment mode, or contiguous bytes pointed to by the LOAD_ADDRESS field otherwise. Poll this bit until it reports DONE to ensure the write is complete.

PKEY_REQUEST_RELOAD: Requests that the private key be requested again from KFUSE. Will autoclear to zero as soon as the requested transfer of the key begins. Only PKEY_LOADED will indicate when the key is ready for use.

PKEY_LOADED: Indicates that the private key value has been received from KFUSE and is ready for use.

LOAD_ADDRESS: For the WRITE16 function, this field selects the start byte address of the contiguous locations in the local key store to be written with the hdcp_key_data. This field is ignored if the AUTOINC bit is ENABLED. Addresses start at 0.

ADDRESS: This read-only field reports the next byte address in the local key store to be written to once the TRIGGER bits are DONE. For the WRITE16 operation, the address will increment by 16. Addresses start at 0.

Typical operation is as follows:

1. Write to the register with LOCAL_ENABLED, AUTOINC_DISABLED, WRITE5_INIT, WRITE7_INIT, LOAD_ADDRESS_ZERO.
2. Write the downstream KSV key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
3. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
4. Write the first downstream key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
5. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
6. Repeat steps 4 and 5 39 more times. There are always 40 keys and one KSV value.

If upstream key authentication is required, do the following:

1. Write the upstream KSV key data into the CD module and decrypt the key.
2. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
3. Write the first upstream key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
4. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
5. Repeat steps 9 and 10 39 more times. There are always 40 keys and one KSV value.

This will leave the store as follows (the required format):

- Byte 0 to Byte 4: downstream KSV
- Byte 5 to Byte 11: 1st downstream key
- Byte 12 to Byte 18: 2nd downstream key

...

- Byte 278 to Byte 284: 40th downstream key
- Byte 285 to Byte 289: upstream KSV
- Byte 290 to Byte 296: 1st upstream key

...

- Byte 563 to Byte 569: 40'th upstream key

Offset: 0xea | Byte Offset: 0x3a8 | Read/Write: R/W | Reset: 0xXXX000X0 (0bxxxxxxxxxx0000000000xxxxxx00xx00)

Bit	R/W	Reset	Description
31:22	RO	X	ADDRESS
21:12	RW	0x0	LOAD_ADDRESS
6	RO	X	PKEY_LOADED: 0 = FALSE 1 = TRUE
5	RW	IDLE	PKEY_REQUEST_RELOAD: 0 = IDLE 1 = TRIGGER
4	RW	DONE	WRITE16: 0 = DONE 1 = TRIGGER 1 = PENDING
1	RW	DISABLED	AUTOINC: 0 = DISABLED 1 = ENABLED
0	RW	DISABLED	LOCAL_KEYS: 0 = DISABLED 1 = ENABLED

27.8.159 SOR_NV_PDISP_KEY_HDCP_KEY_0_0

Offset: 0xee | Byte Offset: 0x3b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

27.8.160 SOR_NV_PDISP_KEY_HDCP_KEY_1_0

Offset: 0xef | Byte Offset: 0x3bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

27.8.161 SOR_NV_PDISP_KEY_HDCP_KEY_2_0

Offset: 0xf0 | Byte Offset: 0x3c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

27.8.162 SOR_NV_PDISP_KEY_HDCP_KEY_3_0

Offset: 0xf1 | Byte Offset: 0x3c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

27.8.163 SOR_NV_PDISP_KEY_HDCP_KEY_TRIG_0

Offset: 0xf2 | Byte Offset: 0x3c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxxx)

Bit	Reset	Description
8	IDLE	LOAD_HDCP_KEY: 0 = IDLE 1 = TRIGGER

27.8.164 SOR_NV_PDISP_KEY_SKEY_INDEX_0

Offset: 0xf3 | Byte Offset: 0x3cc | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	IDX_VALUE: 15 = TEST

27.8.165 SOR_NV_PDISP_SOR_AUDIO_CNTRL0_0

HD Audio (also known as Azalia)

PORT_CONNECTIVITY: This controls the behavior of the Port Connectivity field of the Azalia Configuration Defaults Verb. This can be used to disable a codec that is associated with an SOR that does not connect to a physical port.

- ENABLE: Report 00 in the Port Connectivity field by default. This corresponds to "The Port Complex is connected to a jack"
- DISABLE: Report 01 in the Port Connectivity field by default. This corresponds to "No physical connection for Port" See table 100 and table 101 of the High Definition Audio Specification (Revision 1.0) for more information.

AFIFO_FLUSH: When the DP and HDMI logic are not in the middle of sending an audio packet, they will flush out any entries at the head of the AFIFO that is not tagged as the beginning of a sample. This ensures that the next new audio packet sent will begin on the correct channel. This is a diagnostic workaround bit to disable the flushing.

- ENABLE: Automatically fix the AFIFO if it gets out of alignment
- DISABLE: Do not throw out any AFIFO entries.

SAMPLING_FREQ: This will report the incoming audio stream sampling frequency in the Azalia codec. The HDMI specification only supports 7 audio sample frequencies (fs). Anything other than those 7 fs will be reported as UNKNOWN. See the HDMI 1.1 specification, Table 7-4 (page 77).

SOURCE_SELECT: Determines whether to use the S/PDIF or Azalia (HDAL) audio input. This field has no effect on chips without the HDAL input. The default is AUTO: selects S/PDIF audio until the HDAL audio input is initialized by the external controller

INJECT_NULLSMPL: When the bit is enabled, if the audio format is stereo LPCM as indicated by the stream format in the corresponding output converter widget, the codec inserts null samples into the audio FIFO for each Azalia frame in which it did not receive any samples. This is done only for stereo LPCM and not for any other audio format. This bit should be disabled by default for backwards compatibility.

INPUT_MODE: This bit indicates what the audio data source is. HDA is Azalia data, S/PDIF is S/PDIF data.

SOR_AUDIO_CNTRL0

Offset: 0xfc | Byte Offset: 0x3f0 | Read/Write: R/W | Reset: 0xX01X1000 (0bxx0xxxxxxx01xxxxxxx1xxxxxxxxxx0)

Bit	R/W	Reset	Description
31	RO	X	INPUT_MODE: 0 = HDA 1 = SPDIF
29	RW	DISABLE	INJECT_NULLSMPL: 0 = DISABLE 1 = ENABLE
21:20	RW	SPDIF	SOURCE_SELECT: 0 = AUTO 1 = SPDIF 2 = HDAL

Bit	R/W	Reset	Description
19:16	RO	X	SAMPLING_FREQ: 3 = FREQ_32_0KHZ 0 = FREQ_44_1KHZ 8 = FREQ_88_2KHZ 12 = FREQ_176_4KHZ 2 = FREQ_48_0KHZ 10 = FREQ_96_0KHZ 14 = FREQ_192_0KHZ 1 = FREQ_UNKNOWN
12	RW	ENABLED	AFIFO_FLUSH: 0 = DISABLED 1 = ENABLED
0	RW	ENABLE	PORT_CONNECTIVITY: 0 = ENABLE 1 = DISABLE

27.8.166 SOR_NV_PDISP_SOR_AUDIO_NVAL_0320_0

SOR_AUDIO_NVAL

The following seven registers are used for HDMI N values for Azalia. These registers should be written with the correct N values for the current pixel clock frequency. These values can be found in tables 7-1, 7-2, and 7-3 of the HDMI specification. These registers are initialized to the value in the "other" row of these tables.

For DisplayPort audio, a value of 2^15 (0x8000) is always used.

VALUE: The correct N value for 32 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xff | Byte Offset: 0x3fc | Read/Write: R/W | Reset: 0x00001000 (0bxxxxxxxxxxxxx0000000100000000000)

Bit	Reset	Description
19:0	0x1000	VALUE

27.8.167 SOR_NV_PDISP_SOR_AUDIO_NVAL_0441_0

SOR_AUDIO_NVAL_0441

VALUE: The correct N value for 44.1 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x100 | Byte Offset: 0x400 | Read/Write: R/W | Reset: 0x00001880 (0bxxxxxxxxxxxxx0000000110001000000)

Bit	Reset	Description
19:0	0x1880	VALUE

27.8.168 SOR_NV_PDISP_SOR_AUDIO_NVAL_0882_0

SOR_AUDIO_NVAL_0882

VALUE: The correct N value for 88.2 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x101 | Byte Offset: 0x404 | Read/Write: R/W | Reset: 0x00003100 (0bxxxxxxxxxxxxx0000001100010000000)

Bit	Reset	Description
19:0	0x3100	VALUE

27.8.169 SOR_NV_PDISP_SOR_AUDIO_NVAL_1764_0

SOR_AUDIO_NVAL_1764

VALUE: The correct N value for 176.4 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x102 | Byte Offset: 0x408 | Read/Write: R/W | Reset: 0x00006200 (0bxxxxxxxxxxxxx0000011000100000000)

Bit	Reset	Description
19:0	0x6200	VALUE

27.8.170 SOR_NV_PDISP_SOR_AUDIO_NVAL_0480_0

SOR_AUDIO_NVAL_0480

VALUE: The correct N value for 48 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x103 | Byte Offset: 0x40c | Read/Write: R/W | Reset: 0x00001800 (0bxxxxxxxxxxxxx0000000110000000000)

Bit	Reset	Description
19:0	0x1800	VALUE

27.8.171 SOR_NV_PDISP_SOR_AUDIO_NVAL_0960_0

SOR_AUDIO_NVAL_0960

VALUE: The correct N value for 96 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x104 | Byte Offset: 0x410 | Read/Write: R/W | Reset: 0x00003000 (0bxxxxxxxxxxxxx0000011000000000000)

Bit	Reset	Description
19:0	0x3000	VALUE

27.8.172 SOR_NV_PDISP_SOR_AUDIO_NVAL_1920_0

SOR_AUDIO_NVAL_1920

VALUE: The correct N value for 192 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x105 | Byte Offset: 0x414 | Read/Write: R/W | Reset: 0x00006000 (0bxxxxxxxxxxxxx0000011000000000000)

Bit	Reset	Description
19:0	0x6000	VALUE

27.8.173 SOR_NV_PDISP_SOR_AUDIO_HDA_SCRATCH0_0

SOR_AUDIO_HDA_SCRATCH0/1/2/3

This register is reserved for future use.

Offset: 0x106 | Byte Offset: 0x418 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

27.8.174 SOR_NV_PDISP_SOR_AUDIO_HDA_SCRATCH1_0

Offset: 0x107 | Byte Offset: 0x41c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

27.8.175 SOR_NV_PDISP_SOR_AUDIO_HDA_SCRATCH2_0

Offset: 0x108 | Byte Offset: 0x420 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

27.8.176 SOR_NV_PDISP_SOR_AUDIO_HDA_SCRATCH3_0

Offset: 0x109 | Byte Offset: 0x424 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

27.8.177 SOR_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH0_0

SOR_AUDIO_HDA_CODEC_SCRATCH0/1

These registers are set by the audio driver using vendor-defined verbs. They can be used to pass information from the audio driver to the resource manager if that functionality is ever needed.

When bit 31 changes, an interrupt can be generated (see INT_STATUS register)

Offset: 0x10a | Byte Offset: 0x428 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

27.8.178 SOR_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH1_0

Offset: 0x10b | Byte Offset: 0x42c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

27.8.179 SOR_NV_PDISP_SOR_AUDIO_HDA_ELD_BUFWR_0

SOR_AUDIO_HDA_ELD_BUFWR

Software Programming Model

In an integrated graphics chip, which includes an NVIDIA® Audio Controller, with support for an NVIDIA audio codec driver, the following is the background and the suggested programming model.

The audio software for the HDMI codec will need information about the audio capabilities of an attached HDMI sink device. This information is stored in the HDMI sink device's EDID. Typically, the EDID flows through a graphics adapter to graphics software, so the graphics adapter hardware will not have knowledge of the EDID contents.

To that end, a new mechanism is defined for passing the HDMI sink device's audio EDID information from the graphics software to the audio software. The data payload containing the audio information will be known as EDID-Like Data (or ELD) and will contain a subset of the HDMI sink devices EDID information.

The ELD information will be valid if the HDMI sink is attached and powered on and the ELD Valid bit is set. The Pin Widget that is associated with this HDMI widget will report if the device is attached and that the ELD memory is populated and valid by reporting Presence Detect of 1 and ELD Valid of 1 to a Pin Sense control command. As with the Presence Detect bit, the changes to the ELD Valid bit can also result in the generation of unsolicited responses.

Each codec implements a 96-byte ELD buffer that is written by the resource manager and read by the audio driver. Once the ELD buffer is written by the resource manager, the valid bit, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to

_VALID to indicate that ELD contents have been initialized by the resource manager. On hot unplug events, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to _INVALID to erase ELD programming in the audio driver.

Offset: 0x10c | Byte Offset: 0x430 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	INDEX
7:0	0x0	DATABYTE

27.8.180 SOR_NV_PDISP_SOR_AUDIO_HDA_PRESENSE_0

SOR_AUDIO_HDA_PRESENSE

Reports the Hot Plug state and ELD state to the audio driver. Changes to this state can cause an unsolicited response.

ELDV: Indicates whether the data in the ELD buffer is valid and ready to read.

PD: Presence Detect. This should be set to _PRESENT by software upon hot plug and _NOT_PRESENT on unplug.

Offset: 0x10d | Byte Offset: 0x434 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	INVALID	ELDV: 0 = INVALID 1 = VALID
0	NOT_PRESENT	PD: 0 = NOT_PRESENT 1 = PRESENT

27.8.181 SOR_NV_PDISP_SOR_AUDIO_HDA_CP_0

SOR_AUDIO_HDA_CP

Reports the Content Protection state requested by the Audio driver. It is at the discretion of the video driver to enable or disable content protection. This is a read-only register set by the Content Protection Control (CP_CONTROL) verb.

Offset: 0x10e | Byte Offset: 0x438 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	REQUEST_STATE_VALID
1:0	X	REQUEST_STATE: _DONT_CARE: No state change requested _RESERVED: Unused _PROTECTION_OFF: Audio driver requests content protection to be disabled. _PROTECTION_ON: Audio driver requests content protection to be enabled. 0 = DONT_CARE 2 = PROTECTION_OFF 3 = PTOTECTION_ON

27.8.182 SOR_NV_PDISP_SOR_AUDIO_AVAL_0320_0

AUDIO_AVAL

When using the Azalia codec, it is difficult to recover the 128*fs clock frequency from the incoming audio stream. Therefore, when using the Azalia codec, these registers must be programmed to emulate the N counter.

When using S/PDIF, the N counter will run at 128*fs (audio sampling frequency) and count to a value determined by the HDMI specification. The Azalia counter will run at 24 MHz at all times, and it needs to count for the same period of time as the N counter would have.

Refer to Section 7.2 of the HDMI specification, 1.2.

AVAL values do not need to be programmed by software. Their default values are OK. If the NVAL changes, it will change the N counter frequency. AVAL will need to be changed to match with this new NVAL. Certain resolutions may require different NVALs.

For Mobile chips, the clock is 24 MHz, not 54 MHz. For 44.1, the default N of 6272 gives non-integer AVAL. Choosing N of 4704 instead (nominal CTS 61875) gives an N frequency of 1200 Hz.

48k: 1ms 24000 azaclk cycles (0x5DC0)

44.1k: 1ms/1.2 20000 azaclk cycles (0x4E20)

But this is academic because CTS/N is hardcoded since the pixel clock / audio clock ratios are known and fixed.

At 27 MHz Pixel Clock			
	CTS	**N**	**AVAL**
44.1	22500	4704	20000
88.2	22500	9408	20000
176.4	22500	18816	20000
At 74.25 MHz Pixel Clock			
44.1	61875	4704	20000
88.2	61875	9408	20000
176.4	61875	18816	20000
At 148.5 MHz Pixel Clock:			
44.1	123750	4704	20000
88.2	123750	9408	20000
176.4	123750	18816	20000

VALUE: The correct A value for 32 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x10f | Byte Offset: 0x43c | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx0000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

27.8.183 SOR_NV_PDISP_SOR_AUDIO_AVAL_0441_0

SOR_AUDIO_AVAL_0441

VALUE: The correct A value for 44.1 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x110 | Byte Offset: 0x440 | Read/Write: R/W | Reset: 0x00004e20 (0bxxxxxxxxxxxx0000100111000100000)

Bit	Reset	Description
19:0	0x4e20	VALUE

27.8.184 SOR_NV_PDISP_SOR_AUDIO_AVAL_0882_0

SOR_AUDIO_AVAL_0882

VALUE: The correct A value for 88.2 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x111 | Byte Offset: 0x444 | Read/Write: R/W | Reset: 0x00004e20 (0bxxxxxxxxxxxx0000100111000100000)

Bit	Reset	Description
19:0	0x4e20	VALUE

27.8.185 SOR_NV_PDISP_SOR_AUDIO_AVAL_1764_0

SOR_AUDIO_AVAL_1764

VALUE: The correct A value for 176.4 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x112 | Byte Offset: 0x448 | Read/Write: R/W | Reset: 0x00004e20 (0bxxxxxxxxxxxxx00000100111000100000)

Bit	Reset	Description
19:0	0x4e20	VALUE

27.8.186 SOR_NV_PDISP_SOR_AUDIO_AVAL_0480_0

SOR_AUDIO_AVAL_0480

VALUE: The correct A value for 48 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x113 | Byte Offset: 0x44c | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

27.8.187 SOR_NV_PDISP_SOR_AUDIO_AVAL_0960_0

SOR_AUDIO_AVAL_0960

VALUE: The correct A value for 96 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x114 | Byte Offset: 0x450 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

27.8.188 SOR_NV_PDISP_SOR_AUDIO_AVAL_1920_0

SOR_AUDIO_AVAL_1920

VALUE: The correct A value for 192 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x115 | Byte Offset: 0x454 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

27.8.189 SOR_NV_PDISP_SOR_AUDIO_AVAL_DEFAULT_0

SOR_AUDIO_AVAL_DEFAULT

VALUE: Default A value if the Azalia codec sampling frequency does not match any of the above.

Offset: 0x116 | Byte Offset: 0x458 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

27.8.190 SOR_NV_PDISP_SOR_AUDIO_GEN_CTRL_0

SOR_AUDIO_GEN_CTRL

This register only takes effect when it is written by software. The initialized value is not used.

DEV_ID: Device ID to identify the current chip.

REV_ID: Rev ID for the codec. This value is only used by the codec once this register has been written. Otherwise the default value will be used.

Offset: 0x117 | Byte Offset: 0x45c | Read/Write: R/W | Reset: 0x00290001 (0b0000000000101001xxxxxxxx00000001)

Bit	Reset	Description
31:16	0x29	DEV_ID
7:0	0x1	REV_ID

27.8.191 SOR_NV_PDISP_INT_STATUS_0

Sticky interrupt status, write 1 to clear

Interrupt support: Each interrupt event has 3 configuration states:

MASK	ENABLE	Result
x	0	nothing
0	1	INT_STATUS asserted
1	1	INT_STATUS asserted, and interrupt pin asserted

Offset: 0x11c | Byte Offset: 0x470 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	SCRATCH: SW has written NV_PDISP_SCRATCH register (bit 31 change)
2	X	CP_REQUEST: HDA Codec has written CP_REQUEST register
1	X	CODEC_SCRATCH1: HDA Codec has written CODEC_SCRATCH1 register (bit 31 change)
0	X	CODEC_SCRATCH0: HDA Codec has written CODEC_SCRATCH0 register (bit 31 change)

27.8.192 SOR_NV_PDISP_INT_MASK_0

MASKED prevents the interrupt from asserting the HDMI interrupt pin to the CPU, but still allows the interrupt to appear in INT_STATUS. NOTMASKED allows the interrupt to assert, assuming INT_ENABLE is true also.

Offset: 0x11d | Byte Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	MASKED	SCRATCH_MASK: 0 = MASKED 1 = NOTMASKED
2	MASKED	CP_REQUEST_MASK: 0 = MASKED 1 = NOTMASKED
1	MASKED	CODEC_SCRATCH1_MASK: 0 = MASKED 1 = NOTMASKED
0	MASKED	CODEC_SCRATCH0_MASK: 0 = MASKED 1 = NOTMASKED

27.8.193 SOR_NV_PDISP_INT_ENABLE_0

ENABLE allows the event to appear in INT_STATUS, and allows the interrupt to signal the CPU, assuming INT_MASK=NOTMASKED.

Offset: 0x11e | Byte Offset: 0x478 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	DISABLE	SCRATCH_ENABLE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	DISABLE	CP_REQUEST_ENABLE: 0 = DISABLE 1 = ENABLE
1	DISABLE	CODEC_SCRATCH1_ENABLE: 0 = DISABLE 1 = ENABLE
0	DISABLE	CODEC_SCRATCH0_ENABLE: 0 = DISABLE 1 = ENABLE

27.8.194 SOR_NV_PDISP_SOR_TMDS_HDCP_M0_LO_0

Offset: 0x11f | Byte Offset: 0x47c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	value

27.8.195 SOR_NV_PDISP_SOR_TMDS_HDCP_M0_HI_0

Offset: 0x120 | Byte Offset: 0x480 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	value

27.8.196 SOR_NV_PDISP_SOR_TMDS_HDCP_STATUS_0

Offset: 0x121 | Byte Offset: 0x484 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxx000)

Bit	Reset	Description
8	0x0	OVERRIDE_ENABLE
2	0x0	SCOPE_OVERRIDE
1	0x0	UNPROTECTED_OVERRIDE
0	0x0	RPTR_OVERRIDE

27.8.197 SOR_NV_HDACODEC_AUDIO_GEN_CTL_0

CHSTS_FS_3840

Currently the 4-bit coding for 384 kHz sampling rate is not available in the IEC-61937 specification. If the specification defines this value later, software needs to write the value during device initialization.

COPY_POLARITY: The polarity of the COPY bit is currently inverted in hardware as done in previous Tegra devices.

Offset: 0x122 | Byte Offset: 0x488 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4	OLD	COPY_POLARITY: 0 = OLD 1 = NEW
3:0	0xf	CHSTS_FS_3840

27.8.198 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_CTRL_0

HDMI_VSI_CTRL

This register controls the frequency and generation of the Vendor Specific infoframe packets. The Vendor Specific Infoframe packet contains supplemental information about the raster type. The fields of this register are identical to HDMI_AVI_INFOFRAME_CTRL except where noted below.

The Vendor Specific Infoframe is described in Appendix H of the HDMI 1.4a specification.

- ENABLE

  Setting this field to _YES will initiate infoframe generation. Setting this bit to _NO will disable infoframe generation at the beginning of the next frame.

The frequency of infoframe generation is controlled by OTHER and SINGLE fields.

- OTHER

  Setting this field to _EN while SINGLE is set to _DIS will cause infoframe to be transmitted every other frame.

- SINGLE

  Setting this field to _EN while OTHER is set to _DIS will cause infoframe to be transmitted exactly once.

If OTHER and SINGLE fields are both set to _DIS, infoframe will be generated every frame. Software should never set both OTHER and SINGLE to _EN.

- CHKSUM_HW: Hardware provides a way to calculate the Checksum for the infoframes.
  - _ENABLE will enable the hardware calculation to be passed to the packet
  - _DISABLE will use the register value defined in SF_HDMI_VSI_SUBPACK0_LOW_PB0 for the checksum.
- VIDEO_FMT

  The Vendor Specific Infoframe contains an HDMI_Video_Format Field. This field specifies how the remaining bytes in the infoframe should be interpreted by the specification. When switching to a 3D stereo format, the HDMI_Video_Format field and the 3D_Structure field need to be set appropriately. This priv register field selects whether hardware or software will set those fields.
- _SW_CONTROLLED: The HDMI_Video_Format field and 3D_Structure field will be set by SF_HDMI_VSI_SUBPACK0_HIGH_PB4 and PB5.
- _HW_CONTROLLED: The HDMI_Video_Format field and 3D_Structure field will be set automatically by Hardware based on the state of the NV_917D_Core_Head_SetHdmiCtrl method.

Offset: 0x123 | Byte Offset: 0x48c | Read/Write: R/W | Reset: 0x00010200 (0bxxxxxxxxxxxxxxxx1xxxxxx10xxx0xxx0)

Bit	Reset	Description
16	HW_CONTROLLED	VIDEO_FMT: 0 = SW_CONTROLLED 1 = HW_CONTROLLED
9	ENABLE	CHKSUM_HW: 0 = DISABLE 1 = ENABLE 0 = DIS 1 = EN
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

27.8.199 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_STATUS_0

HDMI_VSI_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in VSI_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0x124 | Byte Offset: 0x490 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 0 = WAITING 1 = DONE

27.8.200 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_HEADER_0

HDMI_VSI_HEADER

This register should be written with the contents of the packet header.

Offset: 0x125 | Byte Offset: 0x494 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

27.8.201 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK0_LOW_0

HDMI_VSI_SUBPACK0_LOW

Bytes 0-3 of the packet are written into this register.

Offset: 0x126 | Byte Offset: 0x498 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

27.8.202 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK0_HIGH_0

HDMI_VSI_SUBPACK0_HIGH

Bytes 4-6 of the packet are written into this register.

Offset: 0x127 | Byte Offset: 0x49c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

27.8.203 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK1_LOW_0

HDMI_VSI_SUBPACK1_LOW

Bytes 7-10 of the packet are written into this register.

Offset: 0x128 | Byte Offset: 0x4a0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

27.8.204 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK1_HIGH_0

HDMI_VSI_SUBPACK1_HIGH

Bytes 11-13 of the packet are written into this register.

Offset: 0x129 | Byte Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

27.8.205 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK2_LOW_0

HDMI_VSI_SUBPACK2_LOW

Bytes 14-17 of the packet are written into this register.

Offset: 0x12a | Byte Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB17
23:16	0x0	PB16
15:8	0x0	PB15
7:0	0x0	PB14

27.8.206 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK2_HIGH_0

HDMI_VSI_SUBPACK2_HIGH

Bytes 18-20 of the packet are written into this register.

Offset: 0x12b | Byte Offset: 0x4ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB20
15:8	0x0	PB19
7:0	0x0	PB18

27.8.207 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK3_LOW_0

HDMI_VSI_SUBPACK3_LOW

Bytes 21-24 of the packet are written into this register.

Offset: 0x12c | Byte Offset: 0x4b0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB24
23:16	0x0	PB23

Bit	Reset	Description
15:8	0x0	PB22
7:0	0x0	PB21

27.8.208 SOR_NV_PDISP_SOR_HDMI_VSI_INFOFRAME_SUBPACK3_HIGH_0

HDMI_VSI_SUBPACK3_HIGH

Bytes 25-27 of the packet are written into this register.

Offset: 0x12d | Byte Offset: 0x4b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB27
15:8	0x0	PB26
7:0	0x0	PB25

27.8.209 SOR_NV_PDISP_SOR_DP_AUDIO_INFOFRAME_HEADER_0

SOR_DP_AUDIO_INFOFRAME_HEADER

This register should be written with the value of the DP Audio InfoFrame header. Audio infoframe is described in CEA-861-D, Section 6.6. The header value written here is overrides the default value (as mentioned in the DP specification) if NV_PDISP_SOR_DP_AUDIO_CTRL_AUDIO_INFOFRAME_HEADER_OVERRIDE is set to ENABLE.

Offset: 0x130 | Byte Offset: 0x4c0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	HB3
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

27.8.210 SOR_NV_PDISP_SOR_DP_AUDIO_INFOFRAME_SUBPACK0_LOW_0

SOR_DP_AUDIO_INFOFRAME_SUBPACK0_LOW

This register should be written with the lower 4 bytes of the DP Audio InfoFrame. PB0, PB1, PB2, and PB3 are defined fields from the specification.

Offset: 0x131 | Byte Offset: 0x4c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

27.8.211 SOR_NV_PDISP_SOR_DP_AUDIO_INFOFRAME_SUBPACK0_HIGH_0

DP_AUDIO_INFOFRAME_SUBPACK0_HIGH

This register should be written with the upper 2 bytes of the DP Audio InfoFrame. PB4 and PB5 are defined fields from the specification.

See Section 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI specification for more information

Offset: 0x132 | Byte Offset: 0x4c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PB5
7:0	0x0	PB4

27.8.212 SOR_NV_PDISP_SOR_DP_AUDIO_TIMESTAMP_0320_0

SOR_DP_AUDIO_TIMESTAMP

Maud and Naud are defined as:

Maud / Naud = (512*fs) / link_clock

In other words, Maud is a measure of how many periods in the 512*fs domain would elapse in Naud link_clocks. Naud is fixed at 2^15. In order to simulate a counter running at 512*fs, a fractional counter running at 108 MHz is used.

For each of these registers, N and D-N should be set such that the following equation is satisfied exactly.

N/D = 512*fs / 108 MHz

32 KHz * 512 = 16.384 MHz

For Tegra X1:

16.384 MHz / 102 MHz = (2^14) / (17 * 3 * 2^4 * 5^3)

= 1024 (N) / 6375 (D)

Offset: 0x133 | Byte Offset: 0x4cc | Read/Write: R/W | Reset: 0x040014e7 (0b00000100000000000000010100111001111)

Bit	Reset	Description
31:16	0x400	N
15:0	0x14e7	D_N

27.8.213 SOR_NV_PDISP_SOR_DP_AUDIO_TIMESTAMP_0441_0

SOR_DP_AUDIO_TIMESTAMP_0441

512 * fs = 512 * 44.1 = 22.5792 MHz

For Tegra X1:

22.5792 / 102 = (2^9 * 3^2 * 7^2) / (17 * 3 * 2^5 * 5^4)

= 2352 (N) / 10625 (D)

Offset: 0x134 | Byte Offset: 0x4d0 | Read/Write: R/W | Reset: 0x09302051 (0b00001001001100000010000001010001)

Bit	Reset	Description
31:16	0x930	N
15:0	0x2051	D_N

27.8.214 SOR_NV_PDISP_SOR_DP_AUDIO_TIMESTAMP_0882_0

SOR_DP_AUDIO_TIMESTAMP_0882

512 * fs = 512 * 88.2 = 45.1584 MHz

For Tegra X1:

45.1584 / 102 = (2^10 * 3^2 * 7^2) / (17 * 3 * 2^5 * 5^4)

= 4704 (N) / 10625 (D)

Offset: 0x135 | Byte Offset: 0x4d4 | Read/Write: R/W | Reset: 0x12601721 (0b00010010011000000001011100100001)

Bit	Reset	Description
31:16	0x1260	N
15:0	0x1721	D_N

27.8.215 SOR_NV_PDISP_SOR_DP_AUDIO_TIMESTAMP_1764_0

SOR_DP_AUDIO_TIMESTAMP_1764

512 * fs = 512 * 176.4 = 90.3168 MHz

For Tegra X1:

90.3168 / 102 = (2^11 * 3^2 * 7^2) / (17 * 3 * 2^5 * 5^4)

= 9408 (N) / 10625 (D)

Offset: 0x136 | Byte Offset: 0x4d8 | Read/Write: R/W | Reset: 0x24c004c1 (0b00100100110000000000010011000001)

Bit	Reset	Description
31:16	0x24c0	N
15:0	0x4c1	D_N

27.8.216 SOR_NV_PDISP_SOR_DP_AUDIO_TIMESTAMP_0480_0

SOR_DP_AUDIO_TIMESTAMP_0480

512 * fs = 512 * 48.0 = 24.576 MHz

For Tegra X1:

24.576 / 102 = (2^13 * 3) / (17 * 3 * 2^4 * 5^3)

= 512 (N) / 2125 (D)

Offset: 0x137 | Byte Offset: 0x4dc | Read/Write: R/W | Reset: 0x0200064d (0b00000010000000000000011001001101)

Bit	Reset	Description
31:16	0x200	N
15:0	0x64d	D_N

27.8.217 SOR_NV_PDISP_SOR_DP_AUDIO_TIMESTAMP_0960_0

SOR_DP_AUDIO_TIMESTAMP_0960

512 * fs = 512 * 96.0 = 49.152 MHz

For Tegra X1:

49.152 / 102 = (2^14 * 3) / (17 * 3 * 2^4 * 5^3)

= 1024 (N) / 2125 (D)

Offset: 0x138 | Byte Offset: 0x4e0 | Read/Write: R/W | Reset: 0x0400044d (0b00000100000000000000010001001101)

Bit	Reset	Description
31:16	0x400	N
15:0	0x44d	D_N

27.8.218 SOR_NV_PDISP_SOR_DP_AUDIO_TIMESTAMP_1920_0

SOR_DP_AUDIO_TIMESTAMP_1920

512 * fs = 512 * 192 = 98.304 MHz

For Tegra X1:

16.384 MHz / 102 MHz = (2^15 * 3) / (17 * 3 * 2^4 * 5^3)

= 2048 (N) / 2125 (D)

Offset: 0x139 | Byte Offset: 0x4e4 | Read/Write: R/W | Reset: 0x0800004d (0b00001000000000000000000001001101)

Bit	Reset	Description
31:16	0x800	N
15:0	0x4d	D_N

27.8.219 SOR_NV_PDISP_HDMI_AUDIO_N_0

AUDIO_N

N_LOOKUP: When set to _ENABLE, the hardware will select the appropriate value of N to use from one of the AUDIO_NVAL registers. This selection is based on the audio sampling frequency detected by the audio block. This is not the sampling frequency reported in the channel status bits. N_RESET should be toggled when this feature is enabled. When set to _DISABLE, software must program the correct N value into AUDIO_N.

Offset: 0x13c | Byte Offset: 0x4f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	0x0	LOOKUP: 1 = ENABLE 0 = DISABLE

27.8.220 SOR_NV_PDISP_HDMI_LANE_CALIB_FUSE_0

Register to read the lane calibration programmed in fuses to get a base line for lane parameters.

Offset: 0x13d | Byte Offset: 0x4f4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	LANE3_CALIB
23:16	X	LANE2_CALIB
15:8	X	LANE1_CALIB
7:0	X	LANE0_CALIB

27.8.221 SOR_NV_PDISP_SOR_HDMI2_CTRL_0

SCRAMBLE: This bit enables scrambling for HDMI 2.0

CLOCK_MODE: This bit allows the clock signal to be divide by 4

SSCP_LENGTH: This sets the length of the SSCP period which is output once per frame. It should not be changed from the default value of 8.

SSCP_START: This sets the start point of the SSCP period. The default setting will output the SSCP period during the VSYNC keepout period.

LANE_SEED: This sets the scrambler reset value for lane. Should not be changed from the default.

Offset: 0x13e | Byte Offset: 0x4f8 | Read/Write: R/W | Reset: 0x02000080 (0b00000010000000000xxxxxxxx1000x000)

Bit	Reset	Description
31:16	0x200	SSCP_START
7:4	0x8	SSCP_LENGTH
2	0x0	SCRAMBLE_AT_LOADV: 0 = DISABLE 1 = ENABLE
1	0x0	CLOCK_MODE: 0 = NORMAL 1 = MODE_DIV_BY_4
0	0x0	SCRAMBLE: 1 = ENABLE 0 = DISABLE

27.8.222 SOR_NV_PDISP_SOR_HDMI2_LFSR0_0

Offset: 0x13f | Byte Offset: 0x4fc | Read/Write: R/W | Reset: 0xfffeffff (0b11111111111111101111111111111111)

Bit	Reset	Description
31:16	0xfffe	LANE1_SEED
15:0	0xffff	LANE0_SEED

27.8.223 SOR_NV_PDISP_SOR_HDMI2_LFSR1_0

Offset: 0x140 | Byte Offset: 0x500 | Read/Write: R/W | Reset: 0x0000fffd (0bxxxxxxxxxxxxxxxx1111111111111101)

Bit	Reset	Description
15:0	0xfffd	LANE2_SEED

27.8.224 SOR_NV_PDISP_SOR_HDCP22_CTRL_0

HDCP 2.2 Control register

This register controls the Encryption of the actual data and initializing the HDCP vars.

_ENABLE: This bit will be set by the software when SKE has successfully completed and it needs to enable the encryption. Setting _YES would actually start encrypting the incoming data.

For HDMI, encryption would start after the EESS has been sent at the beginning of the next frame. For DP, encryption would start at the next SR. Setting this bit to _NO would stop encrypting the data. For HDMI ciphers would stop advancing at next EESS signaling. For DP, cipher would stop at the next CPSR. Hardware will retain internal state. The current status of the CIPHER block is reported in the _CRYPT_STATUS field.

Setting this bit to _NO doesn't reset any of the internal state of Counters so the encryption can be resumed.

_INIT: This bit is set by microcode at the end of the SKE success. Setting this field to _YES would reset inputCtr to its initial value to indicate a fresh start of a new session. Hardware clears this bit as soon as it is done initializing the state and generating the Ciphers.

> ***Note:*** *Software must not toggle this bit when link is already authenticated and encryption has been enabled at least once*

_LOCK_TYPE: This field indicates if the TYPE register can be written by unsecured software or not.

- _LOCKED indicates only Protected software can write to the TYPE
- _UNLOCKED indicates protected software cannot write to the TYPE

  Software needs the init value of the LOCK_TYPE field to be LOCKED.

_REPEATER: This bit is set by microcode normally before triggering the _INIT bit. This bit has no functional impact. It is used in reporting that the SOR is authenticated with a downstream REPEATER to the upstream logic.

_DISABLE_DETACH:HDCP2.2 code is executed in secure code using TSEC in Heavy secure mode. As RM is running in an unsecure environment, there is no "Disable Encryption Command". Currently RM disables HDCP during mode switch and reauthenticates after the mode switch is finished. This field enables HW to disable encryption when SOR is detached from the head. As the SOR is detached, there is no guarantee that the Panel would be in sync with cipher so a new fresh authentication is required. SOFTWARE MUST RESTABLISH THE KS/RIV.

_DISABLE_LANE_CNT:HDCP2.2 code is executed in secure code using TSEC in Heavy secure mode. As RM is running in an unsecure environment, there is no "Disable Encryption Command". Currently RM disables HDCP during Flush mode and reauthenticates after exit from the flush mode. This field enables HW to disable encryption when SOR lane count is set to 0 (which is the first step in Flush mode). As the SOR lane count goes to 0, there is no guarantee that the Panel would be in sync with cipher so a new fresh authentication is required. Software MUST RESTABLISH THE KS/RIV.

Offset: 0x141 | Byte Offset: 0x504 | Read/Write: R/W | Reset: 0x00000064 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx110x100)

Bit	Reset	Description
6	0x1	DISABLE_LANE_CNT0: 0 = YES 1 = NO
5	0x1	DISABLE_DETACH: 0 = YES 1 = NO
4	0x0	REPEATER: 1 = YES 0 = NO
2	0x1	LOCK_TYPE: 0 = UNLOCKED 1 = LOCKED
1	0x0	INIT: 0 = DONE 1 = TRIGGER
0	0x0	CRYPT: 0 = DISABLE 1 = ENABLE

27.8.225 SOR_NV_PDISP_SOR_HDCP22_STATUS_0

HDCP 2.2 Status register

This register provides the status of the HDCP2.2 block.

_CRYPT_STATUS: This bit reports the actual link encryption status. This bit indicates whether HW is advancing the cipher or not.

_FRAME_CNT_OVERFLOW: This indicates if the Frame counter has overflowed.

_DATA_CNT_OVERFLOW: This indicates if the Data counter has overflowed.

_DETACHED_DISABLE: When NV_PDISP_SOR_HDCP22_CTRL_DISABLE_DETACH is set, HW would disable encryption when the SOR is detached. The status is reported in this bit. This will also indicate CRYPT_STATUS to _INACTIVE.

_LANE_CNT0_DISABLE: When NV_PDISP_SOR_HDCP22_CTRL_DISABLE_LANE_CNT0 is set, HW would disable encryption when the SOR lane count is set to 0 in DP mode. The status is reported in this bit. This will also indicate CRYPT_STATUS to _INACTIVE

Offset: 0x142 | Byte Offset: 0x508 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:9	X	HDCP_STATE: 0 = IDLE 1 = WAIT_LC128 2 = WAIT_AES_READY 3 = HDCP22_ENABLE 4 = HDMI_ENCRYPT_ON 5 = DP_ENCRYPT_ON
8:7	X	AUTODIS_STATE: 0 = IDLE 1 = ENCRYPTING 2 = DISABLE_LC_0 3 = DISABLE_DETACH
6	X	LC128_ERROR: 0 = NO 1 = YES
5	X	LANE_CNT0_DISABLE: 0 = NO 1 = YES
4	X	DETACHED_DISABLE: 0 = NO 1 = YES
3	X	DATA_CNT_OVERFLOW: 0 = NO 1 = YES
2	X	FRAME_CNT_OVERFLOW: 0 = NO 1 = YES
0	X	CRYPT_STATUS: 0 = INACTIVE 1 = ACTIVE

27.8.226 SOR_NV_PDISP_SOR_HDCP22_AES_CTR_KEY_MSB_0

AES-CTR Key bus

Next 4 registers holds the Key value to drive the AES-CTR. This register will be programmed by SW to Session key value. This value will be XORed with the lc128. This register must be set to correct Ks value before setting the NV_PDISP_SOR_HDCP22_CTRL_ENABLE to _YES.

Offset: 0x144 | Byte Offset: 0x510 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.227 SOR_NV_PDISP_SOR_HDCP22_AES_CTR_KEY_LSB1_0

Offset: 0x145 | Byte Offset: 0x514 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.228 SOR_NV_PDISP_SOR_HDCP22_AES_CTR_KEY_LSB2_0

Offset: 0x146 | Byte Offset: 0x518 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.229 SOR_NV_PDISP_SOR_HDCP22_AES_CTR_KEY_LSB3_0

Offset: 0x147 | Byte Offset: 0x51c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.230 SOR_NV_PDISP_SOR_HDCP22_AES_CTR_DATA_MSB_0

AES-CTR Data bus

Next 2 registers hold the Data input value to drive the AES-CTR. This register will be programmed by SW to 64 bits riv value. For HDMI, this value will be concatenated by inputCtr value for specific protocol. For DP, this value will be XORed with TYPE value. This register must be set to correct riv value before setting the NV_PDISP_SOR_HDCP22_CTRL_ENABLE to _YES.

Offset: 0x148 | Byte Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.231 SOR_NV_PDISP_SOR_HDCP22_AES_CTR_DATA_LSB_0

Offset: 0x149 | Byte Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.232 SOR_NV_PDISP_SOR_HDCP22_SST_DP_TYPE_0

AES-CTR DP Type

Below 2 registers are used when the SOR is set to DP protocol. It allows SW to pass the Stream Type value to the HW which is used during encryption. This is a 64 bits value and indicates the type of the stream. If any bit is set to 1, then corresponding stream cannot be transmitted to HDCP1.x devices.

This communication is handled by microcode as a part of the Stream_Manage_Message to the repeater.

Hardware uses this value to generate the ciphers.

New settings in these registers are promoted to active at the same time as NV_PDISP_SOR_ECF0/1.

Offset: 0x14a | Byte Offset: 0x528 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE

27.8.233 SOR_NV_PDISP_SOR_HDCP22_LC128_MSB_0

LC128- Global HDCP constant used for the cipher computation in HDCP hardware.

Offset: 0x14b | Byte Offset: 0x52c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.234 SOR_NV_PDISP_SOR_HDCP22_LC128_LSB1_0

Offset: 0x14c | Byte Offset: 0x530 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.235 SOR_NV_PDISP_SOR_HDCP22_LC128_LSB2_0

Offset: 0x14d | Byte Offset: 0x534 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.8.236 SOR_NV_PDISP_SOR_HDCP22_LC128_LSB3_0

Offset: 0x14e | Byte Offset: 0x538 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

27.9 HDCP KFUSE Control Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

27.9.1 KFUSE_STATE_0

Commands and status for ECC mode

Offset: 0x80 I Read/Write: R/W I Reset: 0xXX0XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	X	SOFTRESET
25	RW	X	STOP: Write 1 to abort decoding in progress, then wait for STATE=IDLE
24	RW	X	RESTART: Write 1 to re-start decoding, similar to deassertion of reset
17	RO	X	CRCPASS: After DONE, indicates CRC pass/fail
16	RO	X	DONE: When decode is complete, DONE=1
13:8	RO	X	ERRBLOCK: If any ERR_* are set, contains offset of first errored block
5:0	RO	X	CURBLOCK: Counter of current block during decode, for debugging

27.9.2 KFUSE_ERRCOUNT_0

ECC decode error count; valid after DONE=1

Offset: 0x84 I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:24	X	ERR_FATAL: number of uncorrectable errors
22:16	X	ERR_3: number of correctable 3-bit errors
14:8	X	ERR_2: number of correctable 2-bit errors
6:0	X	ERR_1: number of correctable 1-bit errors

27.9.3 KFUSE_KEYADDR_0

For reading keyglob data after decode is DONE

Offset: 0x88 I Read/Write: R/W I Reset: 0x00010000 (0bxxxxxxxxxxxxxxx1xxxxxxx00000000)

Bit	Reset	Description
16	0x1	AUTOINC: when set, ADDR is incremented by 1 after each read of KEYS
7:0	0x0	ADDR: Word address (0..144)

27.9.4 KFUSE_KEYS_0

Decoded keyglob data; after each read, ADDR is incremented if AUTOINC==1

Offset: 0x8c I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

27.9.5 KFUSE_PD_0

Offset: 0x24 I Read/Write: R/W I Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	R/W	Reset	Description
1	RO	X	STATUS:PD STATUS BIT: Indicates the final value of PD going to the fuse macro. 0 = KFUSECELL_PWRUP 1 = KFUSECELL_PWRDOWN
0	RW	0x0	CTRL:PD (or POWER DOWN) feature: Writing this bit will assert kfusecell_pd pin to kfusecell macro and put it in a low leakage mode. 0 = POWERUP 1 = POWERDOWN

27.10 HDA Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

27.10.1 HDA_AXI_BAR0_SZ_0

AXI BARi mapping

Offset: 0x0 I Read/Write: R/W I Reset: 0x00000004 (0bxxxxxxxxxxxxx00000000000000000100)

Bit	Reset	Description
19:0	0x4	AXI_BAR0_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

27.10.2 HDA_AXI_BAR1_SZ_0

Offset: 0x4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR1_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

27.10.3 HDA_AXI_BAR2_SZ_0

Offset: 0x8 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR2_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

27.10.4 HDA_AXI_BAR3_SZ_0

Offset: 0xc I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR3_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

27.10.5 HDA_AXI_BAR0_START_0

Offset: 0x40 I Read/Write: R/W I Reset: 0x70038000 (0b01110000000000111000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x70038	AXI_BAR0_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

27.10.6 HDA_AXI_BAR1_START_0

Offset: 0x44 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR1_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

27.10.7 HDA_AXI_BAR2_START_0

Offset: 0x48 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR2_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

27.10.8 HDA_AXI_BAR3_START_0

Offset: 0x4c I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR3_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

27.10.9 HDA_FPCI_BAR0_0

Offset: 0x80 I Read/Write: R/W I Reset: 0x00040001 (0b00000000000001000000000000000xxx1)

Bit	Reset	Description
31:4	0x4000	FPCI_BAR0_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this register.
0	0x1	FPCI_BAR0_ACCESS_TYPE:Indicates if the address region is memory mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=I/O/config access (NPW only)

27.10.10 HDA_FPCI_BAR1_0

Offset: 0x84 I Read/Write: R/W I Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR1_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this register.
0	0x1	FPCI_BAR1_ACCESS_TYPE:Indicates if the address region is memory mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=I/O/config access (NPW only)

27.10.11 HDA_FPCI_BAR2_0

Offset: 0x88 I Read/Write: R/W I Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR2_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a a shift left 12 of the value of this register.
0	0x1	FPCI_BAR2_ACCESS_TYPE:Indicates if the address region is memory mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=I/O/config access (NPW only)

27.10.12 HDA_FPCI_BAR3_0

Offset: 0x8c I Read/Write: R/W I Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR3_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this register.
0	0x1	FPCI_BAR3_ACCESS_TYPE:Indicates if the address region is memory mapped versus configuration or I/O space. 0=memory mapped access (PW only) 1=I/O/config access (NPW only)

27.10.13 HDA_MSI_BAR_SZ_0

MSI BAR SIZE

Offset: 0xc0 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE: The size of the address range associated with MSI BAR is in 4K increments. Value of 0 signifies BAR is not used.

27.10.14 HDA_MSI_AXI_BAR_ST_0

MSI AXI BAR START

Offset: 0xc4 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	MSI_AXI_BAR_START: The start of upstream AXI address space for MSI BAR.The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

27.10.15 HDA_MSI_FPCI_BAR_ST_0

MSI FPCI BAR START

Offset: 0xc8 I Read/Write: R/W I Reset: 0x00000000 (0b0000000000000000000000000000xxxx)

Bit	Reset	Description
31:4	0x0	MSI_FPCI_BAR_START: The start of upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

27.10.16 HDA_MSI_VEC0_0

MSI VECTORi i in [0 7] RW

Offset: 0x100 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR0: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.17 HDA_MSI_VEC1_0

Offset: 0x104 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR1: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.18 HDA_MSI_VEC2_0

Offset: 0x108 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR2: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.19 HDA_MSI_VEC3_0

Offset: 0x10c I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR3: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.20 HDA_MSI_VEC4_0

Offset: 0x110 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR4: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.21 HDA_MSI_VEC5_0

Offset: 0x114 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR5: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.22 HDA_MSI_VEC6_0

Offset: 0x118 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR6: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.23 HDA_MSI_VEC7_0

Offset: 0x11c I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR7: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

27.10.24 HDA_MSI_EN_VEC0_0

MSI ENABLE VECTORi□i in [0□7]□RW

Offset: 0x140 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR0: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.25 HDA_MSI_EN_VEC1_0

Offset: 0x144 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR1: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.26 HDA_MSI_EN_VEC2_0

Offset: 0x148 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR2: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.27 HDA_MSI_EN_VEC3_0

Offset: 0x14c I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR3: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.28 HDA_MSI_EN_VEC4_0

Offset: 0x150 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR4: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.29 HDA_MSI_EN_VEC5_0

Offset: 0x154 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR5: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.30 HDA_MSI_EN_VEC6_0

Offset: 0x158 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR6: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.31 HDA_MSI_EN_VEC7_0

Offset: 0x15c I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR7: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

27.10.32 HDA_CONFIGURATION_0

Configuration

Offset: 0x180 I Read/Write: R/W I Reset: 0x800X8X40 (0b1xxxxxxxxxxx1xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x1	CLKEN_OVERRIDE: This can override the clock enable in case of malfunction.
19	RW	0x1	PW_NO_DEVSEL_ERR_CYA: Setting this bit disables detection of DECERR due to no DEVSEL for DS PWs only.
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on AFI upstream A value of 1b indicates there are no outstanding reads to initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on AFI upstream A value of 1b indicates there are no outstanding writes to initiator.
15	RW	0x1	WDATA_LEAD_CYA: Used to enable/disable the handling of write data ahead of requests on IPFS axi target
14	RW	0x0	WR_INTRLV_CYA: Used to enable/disable the handling of interleaved write requests on IPFS axi target
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to IPFS target. A value of 1b indicates there are no outstanding reads to downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to IPFS target. A value of 1b indicates there are no outstanding writes to downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any active bits valid or not

Bit	R/W	Reset	Description
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = whenever MSI is ready assert the interrupt 0 = default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: input to upstream FPCI 1 = whenever write is ready, send it; 0 = write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to upstream FPCI. Allow upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: used for upstream FPCI.Allow upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: used for downstream FPCI.Allow downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: input to downstream FPCI.Allow downstream FPCI responses to pass writes
1	RW	0x0	DFPCI_PASSPW: input to downstream FPCI.Allow downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the IPFS device block is disabled, it is completely invisible on the IPFS bus, i.e., it doesn't even process IPFS configuration accesses.

27.10.33 HDA_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0x184 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Master Abort. 1 = forward error, 0 = return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Data Error. 1 = forward error, 0 = return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows FPCI error to be forwarded to AXI response when FPCI error response. This bit also covers decode error generated when there is no devsel received indicates Target Abort. 1 = forward error, 0 = return AXI OKAY response (2'b0)

27.10.34 HDA_INTR_MASK_0

Interrupt Masks

Offset: 0x188 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxx0xxxxxxx0)

Bit	Reset	Description
16	0x0	IP_INT_MASK:IP (SATA/AZA) interrupt to MPCORE gated by mask.
8	0x0	MSI_MASK:MSI to MPCORE gated by mask.
0	0x0	INT_MASK:Interrupt to MPCORE gated by mask.

27.10.35 HDA_INTR_CODE_0

Interrupt Control

Offset: 0x18c I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	INT_CODE:Eight interrupt codesIf the code is 0, logging of the next interrupt is enabled 0 = INT_CODE_CLEAR: Clear interrupt code 1 = INT_CODE_INI_SLVERR: Interrupt code for MPCORE AXI SLVERR response to IPFS 2 = INT_CODE_INI_DECERR: Interrupt code for MPCORE AXI DECERR response to IPFS 3 = INT_CODE_TGT_SLVERR: Interrupt code for PCIE endpoint FPCI target abort or data error response to IPFS 4 = INT_CODE_TGT_DECERR: Interrupt code for PCIE2 FPCI master abort response to IPFS 5 = INT_CODE_TGT_WRERR: Interrupt code for bufferable write to non-posted write address region 6 = RSVD1: Reserved 7 = INT_CODE_DFPCI_DECERR: Interrupt code for PCIE2 response to downstream request when downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR: Interrupt code for IPFS response to downstream request when axi target AXI addresss does not fall in any of IPFS downstream BARs 9 = INT_CODE_FPCI_TIMEOUT: Interrupt code for FPCI Timeout 10 = RSVD2: Reserved for future expansion 11 = RSVD3 12 = RSVD4 13 = RSVD5 14 = RSVD6 15 = INT_CODE_SM_FATAL_ERROR: Interrupt code for SM fatal error 16 = INT_CODE_SM_NON_FATAL_ERROR: Interrupt code for SM non-fatal error

27.10.36 HDA_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0x190 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, it contains address bits [31:2], either in FPCI memory space or AXI space. For FPCI generated errors, the info contains FPCI address.For AXI/IPFS generated errors, the info contains AXI address.
0	0x0	DIR: Indicates direction of the AXI/FPCI transaction. 1=rd/0=wrIf signature type is 6 (sideband message), this field is 1. 0 = WRITE: Interrupt due to a write transaction 1 = READ: Interrupt due to a read transaction

27.10.37 HDA_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0x194 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of captured FPCI address (bits [39:32])when interrupt code is 3, 4 or 7. These bits determine the region in the Hypertransport Address Map that was accessed.

27.10.38 HDA_IPFS_INTR_ENABLE_0

IPFS Interrupt Enable

Offset: 0x198 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxx00000000)

Bit	Reset	Description
13	0x0	EN_SM_NON_FATAL_ERROR: Enable bit for interrupt code 15
12	0x0	EN_SM_FATAL_ERROR: Enable bit for interrupt code 14
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5

Bit	Reset	Description
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

27.10.39 HDA_UFPCI_CONFIG_0

Upstream FPCI Configuration

Offset: 0x19c I Read/Write: R/W I Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00010)

Bit	Reset	Description
4:0	0x2	UNITID_T0C0: Upstream FPCI Unit ID for controller 0. HyperTransport, upstream FPCI request

27.10.40 HDA_CFG_REVID_0

CFG_REVID register

Offset: 0x1a0 I Read/Write: R/W I Reset: 0x000X10XX (0bxxxxxxxxxxxxxxxxxx0100xxxxxxx1xx)

Bit	R/W	Reset	Description
19	RO	X	DEV2SM_NONISO_REQUEST_PEND: This is to tell if there is a non ISO request pending. 0 = NO 1 = YES
18	RO	X	DEV2SM_ISO_REQUEST_PEND: This is to tell if there is an ISO request pending. 0 = NO 1 = YES
13:12	RW	0x1	STRAP_CPU_MODE: MCP: mode to send MSI. Can have it programmable 0 = NB_INTEL 1 = NB_AMD 2 = AMD 3 = TMTA
11	RW	0x0	CFG_REVID_WRITE_ENABLE: MCP: the enable to override the revid. Can have it programmable 0 = CLEAR 1 = SET
10	RW	0x0	CFG_REVID_OVERRIDE: MCP: a way to override the current revision ID. Can have it programmable 0 = DISABLE 1 = ENABLE
4	RO	X	DEV2LEG_NONCOH_REQUEST_PEND: MCP: Tells the leg block that we have a non coherent req pending. 0 = NO 1 = YES
3	RO	X	DEV2LEG_COH_REQUEST_PEND: MCP comment: Tells the leg block that we have a coherent req pending 0 = NO 1 = YES
2	RW	0x1	SM2DEV_FPCI_TIMEOUT_EN: fpci timeout enable bit for Controller 0: disable 1: enable 0 = DISABLE 1 = ENABLE

27.10.41 HDA_FPCI_TIMEOUT_0

FPCI_TIMEOUT register

Offset: 0x1a4 I Read/Write: R/W I Reset: 0x000f0000 (0bxxxxxxxxxxxxx11110000000000000000)

Bit	Reset	Description
19:0	0xf0000	SM2ALL_FPCI_TIMEOUT_THRESH: This sets the timeout thresh value for fpci bus. The starts counting for each queues (iso/niso- rd/wr) have pending request in fpci wrapper, the count resets when the requests popped

27.10.42 HDA_TOM_0

Top of Memory Limit

Offset: 0x1a8 I Read/Write: R/W I Reset: 0x3fff0fff (0bxx11111111111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3fff	LEG2ALL_TOM2: Top of Memory Limit 2.
11:0	0xfff	LEG2ALL_TOM1: Top of Memory Limit 1.

27.10.43 HDA_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x1ac I Read/Write: RO I Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND: Number of pending initiator iso PW responses

27.10.44 HDA_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x1b0 I Read/Write: RO I Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND: Number of pending initiator niso PW responses

27.10.45 HDA_INTR_STATUS_0

IPFS Interrupt Status

Offset: 0x1b4 I Read/Write: RO I Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	IP_INTR_STATUS: status of IP (SATA/AZA) interrupt
1	X	MSI_INTR_STATUS: status of MSI interrupt
0	X	IPFS_INTR_STATUS: status of ipfs interrupt

27.10.46 HDA_DFPCI_BEN_0

Downstream FPCI Byte Enables

Offset: 0x1b8 I Read/Write: R/W I Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	0x0	EN_DFPCI_BEN: Enable bit for ben; when set, programmed be is sent on DFPCI bus
3:0	0x0	DFPCI_BYTE_ENABLE_N: Active low byte enables

27.10.47 HDA_CLKGATE_HYSTERESIS_0

Offset: 0x1bc I Read/Write: R/W I Reset: 0x00000014 (0bxxxxxxxxxxxxxxxxxxxxxxxx00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of IPFS clock cycles to wait after clock gating criteria is met to disable IPFS/FPCI clocks

27.10.48 HDA_SPARE_REG0_0

Offset: 0x1d8 I Read/Write: R/W I Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:0	-65536	IPFS_SPARE_REG: Spare Register

27.10.49 HDA_HDA_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register.

Note: *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility. The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and mc side of the mccif. All clock gating is enabled by default.*

A '1' written to the rclk/wclk override field will result in one of the following:

With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clk is enabled. With wclk/rclk override mode = ON, the clock is always on inside mccif and PC.

A '1' written to the cclk override field keeps client's clk always on inside mccif.

Offset: 0x1dc I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	HDA_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	HDA_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	HDA_CCLK_OVERRIDE
17	0x0	HDA_RCLK_OVERRIDE
16	0x0	HDA_WCLK_OVERRIDE
3	DISABLE	HDA_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	HDA_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	HDA_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	HDA_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

27.10.50 HDA_MISC_0

Offset: 0x1e0 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	HDA_DEVICE_DIS: Serial ATA Interface 0 Disable 1 = Azalia Interface 0 Disabled (Not seen as part of PCI space) 0 = Azalia Interface 0 Enabled. 0 = ENABLE 1 = DISABLE

27.10.51 HDA_ORDERING_RULES_0

Offset: 0x1e4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000x)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW: Modified Upstream MSIAW ordering. 0 = MSIAW behavior 1 = Legacy (Tegra 3) MSIAW behavior
2	0x0	UPSTREAM_RESPAW: Modified RespAW ordering. 0 = RespAW behavior 1 = Legacy (Tegra 3) RespAW behavior
1	0x0	UPSTREAM_RAW: Modified RAW ordering. 0 = RAW behavior 1 = Legacy (Tegra 3) RAW behavior

27.10.52 HDA_A2F_UFPCI_CFG0_0

Offset: 0x1e8 I Read/Write: R/W I Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

27.10.53 HDA_A2F_UFPCI_CFG1_0

Offset: 0x1ec I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

27.10.54 HDA_DUMMY_REG_0

Dummy register to get the last valid register address

Offset: 0x1f0 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	DUMMY:Dummy register

27.11 DPAUX Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

27.11.1 DPAUX_CTXSW_0

Context Switch Register

Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be preloaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

27.11.2 DPAUX_INTR_EN_AUX_0

A DP SOR port can generate 4 types of interrupts:

- PLUG_EVENT: The HPD line has been high for at least DPAUX_HPD_CONFIG_0_PLUG_MIN_TIME microseconds
- UNPLUG_EVENT: The HPD line has gone low for at least DPAUX_HPD_CONFIG_0_UNPLUG_MIN_TIME microseconds
- IRQ_EVENT: After a DP device has been connected, it can generate an interrupt request. This will cause the DP hardware to read the Sink status and then notify the RM with an IRQ_EVENT interrupt.
- AUX_DONE: This interrupt will signify the completion of an AUX transaction.
- This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x1..0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	DISABLED	AUX_DONE: 0 = DISABLED 1 = ENABLED
2	DISABLED	IRQ_EVENT: 0 = DISABLED 1 = ENABLED
1	DISABLED	UNPLUG_EVENT: 0 = DISABLED 1 = ENABLED
0	DISABLED	PLUG_EVENT: 0 = DISABLED 1 = ENABLED

27.11.3 DPAUX_INTR_AUX_0

The interrupt status generated by the DP SOR over AUX logic can be determined by reading this register. A pending interrupt can be cleared by writing 1 to the bit associated with that interrupt.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x5..0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	NOT_PENDING	AUX_DONE: 0 = NOT_PENDING 1 = PENDING
2	NOT_PENDING	IRQ_EVENT: 0 = NOT_PENDING 1 = PENDING
1	NOT_PENDING	UNPLUG_EVENT: 0 = NOT_PENDING 1 = PENDING
0	NOT_PENDING	PLUG_EVENT: 0 = NOT_PENDING 1 = PENDING

27.11.4 DPAUX_DP_AUXDATA_WRITE_W0_0

SOR_DP_AUXDATA_WRITE_W0 (REG_0)
SOR_DP_AUXDATA_WRITE_W1 (REG_1)
SOR_DP_AUXDATA_WRITE_W2 (REG_2)
SOR_DP_AUXDATA_WRITE_W3 (REG_3)

Data register array for DisplayPort writes.

All 4 registers are combined into a 16-byte data buffer. This buffer contains data portion of an AUX native write, as specified in the DisplayPort specification. An AUX native reads maximum 16 bytes of data. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x9..0xc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

27.11.5 DPAUX_DP_AUXDATA_WRITE_W1_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0xd..0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

27.11.6 DPAUX_DP_AUXDATA_WRITE_W2_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x11..0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

27.11.7 DPAUX_DP_AUXDATA_WRITE_W3_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x15..0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REG

27.11.8 DPAUX_DP_AUXDATA_READ_W0_0

DP_AUXDATA_READ_W0 (REG_0)
DP_AUXDATA_READ_W1 (REG_1)
DP_AUXDATA_READ_W2 (REG_2)
DP_AUXDATA_READ_W3 (REG_3)

Data register array for DisplayPort reads.

All 4 registers are combined into a 16-byte data buffer. This buffer contains data portion of an AUX native read, as specified in the DisplayPort specification. An AUX native reads maximum 16 bytes of data. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x19..0x1c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

27.11.9 DPAUX_DP_AUXDATA_READ_W1_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x1d..0x20 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

27.11.10 DPAUX_DP_AUXDATA_READ_W2_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x21..0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

27.11.11 DPAUX_DP_AUXDATA_READ_W3_0

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x25..0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

27.11.12 DPAUX_DP_AUXADDR_0

DP_AUXADDR

Address register for DisplayPort AUX channel transaction.

This address register describes the 20-bit address that the transaction reads from/writes to the sink AUX register address space. Refer to the DisplayPort specification for more details.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x29..0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	REG

27.11.13 DPAUX_DP_AUXCTL_0

DP_AUXCTL

Main control register for AUX transactions.

Refer to the DisplayPort specification for more information about AUX channel transactions.

The AUX bus can transmit data at 1Mbps. A single AUX transaction may take up to 648 microseconds to complete:

- 66 µs for precharge, SYNC, COMMAND, ADDRESS, and LENGTH
- 16*8 µs DATA
- 2 µs SYNC/STOP
- 400 µs maximum delay before reply begins
- 34 µs for precharge, SYNC
- 8 µs for REPLY_TYPE
- 8 µs LENGTH
- 2 µs SYNC/STOP

  ----

  648 microseconds

If an HPD_IRQ arrives just before an AUX request occurs, hardware will use the AUX channel to read the Link/Sink status, and then process the software request. This will incur an additional delay of up to 568 microseconds before the software request begins. This brings the total worst-case reply time to 1216 microseconds.

The AUXCTL_SEMA_REQUEST and AUXCTL_SEMA_GRANT bits are the semaphore for the DisplayPort AUX channel. Both VBIOS and RM need to use the AUX channel. The AUX semaphore is requested by writing to AUXCTL_SEMA_REQUEST. The client must check the value of AUXCTL_SEMA_GRANT to see if it was awarded the semaphore.

To obtain the semaphore:

For RM:

```
Write _RM to AUXCTL_SEMA_REQUEST
read AUXCTL_SEMA_GRANT
if (AUXCTL_SEMA_GRANT == _RM)
     RM grabbed the AUX channel successfully
else
     RM failed to control the AUX
```

For VBIOS:

```
Write _VBIOS to AUXCTL_SEMA_REQUEST
read AUXCTL_SEMA_GRANT;
if (AUXCTL_SEMA_GRANT == _VBIOS)
     VBIOS grabbed the AUX channel successfully;
else
     VBIOS failed to control the AUX;
```

To release the semaphore:

After the AUX transaction is finished, software must write _RELEASE to SEMA_REQUEST. This can be used to put the AUX channel back into a good state if it gets corrupted.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x2d..0x30 | Read/Write: R/W | Reset: 0x0X000000 (0b0xxxxxxxxx00xxx00000xxx000000000)

Bit	R/W	Reset	Description
31	RW	DEASSERT	RST: This can be used to put the AUX channel back into a good state if it gets corrupted. Set to ASSERT to force a localized reset of the AUX logic. Set to DEASSERT to allow the AUX engine to run 0 = DEASSERT 1 = ASSERT
25:24	RO	X	SEMA_GRANT: The client must check the value of AUXCTL_SEMA_GRANT to see if it was awarded the semaphore. SEMA_GRANT_NONE: AUX channel available SEMA_GRANT_RM: AUX channel obtained by RM SEMA_GRANT_VBIOS: AUX channel obtained by VBIOS SEMA_GRANT_PMU: AUX channel obtained by PMU 0 = NONE 1 = RM 2 = VBIOS 3 = PMU
21:20	RW	RELEASE	SEMA_REQUEST: The AUX semaphore is requested by writing to AUXCTL_SEMA_REQUEST. The process for RM or VBIOS to obtain the AUX channel is: Set to _RM if aux is obtained by RM. Set to _VBIOS if aux is obtained by VBIOS. Set to RELEASE to relinquish control of the AUX channel. 0 = RELEASE 1 = RM 2 = VBIOS 3 = PMU
16	RW	DONE	TRANSACTREQ: Software uses this bit to request an AUX channel transaction. After CMD, CMDLEN, and AUXDATA_WRITE have been set, this field should be set to _TRIGGER (1). After the hardware receives the reply, the hardware clears the TRANSACTREQ to _DONE (0). If an unplug event is detected in the middle of a transaction, the current transaction is aborted, and the UNPLUG_EVENT bit will be set in DPAUX_INTR_4. Further transactions while unplugged will still be sent. If DPAUX_HYBRID_PADCTL_MODE is set to _I2C, or DPAUX_HYBRID_SPARE_PAD_PWR is set to _POWERDOWN, no transactions will be sent out. Any transactions in flight will complete. This prevents any interference with the I2C functionality of the pad. At the completion of an AUX transaction, the hardware will write the results into the DPAUX_DP_AUXSTAT register. The reply type should be checked as well as the error bits in this register to determine if the request was successful 0 = DONE 1 = PENDING
15:12	RW	I2CWR	CMD: AUX Command. Refer to Section 2.4, AUX channel syntax, in the DisplayPort specification. I2CWR: I2C write I2CRD: I2C read I2CREQWSTAT: I2C write status request I2CMOTWR: I2C write, Middle of transaction (MOT) I2CMOTRD: I2C read, MOT MOTREQWSTAT: I2C write status request, (MOT) AUXWR: AUX write AUXRD: AUX read 0 = I2CWR 1 = I2CRD 2 = I2CREQWSTAT 4 = MOTWR 5 = MOTRD 6 = MOTREQWSTAT 8 = AUXWR 9 = AUXRD

Bit	R/W	Reset	Description
8	RW	NO	ADDRESS_ONLY: Some I2C-over-AUX operations require address-only transactions. This bit modifies any AUX transaction into an address only transaction. The DP specification only uses Address-only transactions for I2CRD and I2CWR (with or without MOT set). NO: Send the AUX transaction normally. YES: Send an Address-only AUX transaction. Only Command and Address will be sent. The CMDLEN field is ignored; no data will be written. Since no data is written, the ACK/NACK for this command will not contain an M value. The values in DPAUX_DP_AUXSTAT_REPLY_M and DPAUX_DP_AUXSTAT_RX_ERROR will not be accurate after an Address-only transaction. 0 = NO 1 = YES
7:0	RW	0x0	CMDLEN: These bits determine the number of data bytes an AUX channel transaction writes or reads. An AUX transaction can read or write a maximum 16 bytes. Note: This field should be set to N-1, where N is the number of bytes in the transaction.

27.11.14 DPAUX_DP_AUXSTAT_0

DP_AUXSTAT

When an AUX command completes, it will update most fields in this register.

Refer to the DisplayPort specification for more information about AUX channel transaction.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x31..0x34 | Read/Write: R/W | Reset: 0xX0XX00XX (0bxxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

Bit	R/W	Reset	Description
28	RO	X	HPD_STATUS: Reports the current state of the HPD line, plugged or unplugged. This field is valid at all times, not just at the end of a transaction. This status is based on the value of the GPIO associated with this AUX channel. The connection between GPIO and AUX channel is hard coded, so they must be kept together when assigning them to an SOR link. AUX0 GPIO(1) AUX1 GPIO(19) AUX2 GPIO(15) AUX3 GPIO(21) 0 = UNPLUG 1 = PLUGGED
23:20	RO	X	AUXCTL_STATE: Reports the current state of the AUXCTL state machine. This field is valid at all times, not just at the end of a transaction. 0 = IDLE 1 = SYNC 2 = START1 3 = COMMAND 4 = ADDRESS 5 = LENGTH 6 = WRITE1 7 = READ1 8 = GET_M 9 = STOP1 10 = STOP2 11 = REPLY 12 = CLEANUP

Bit	R/W	Reset	Description
19:16	RO	X	REPLYTYPE: These bits are for status and describe the reply type of the AUX channel command. Refer to Section 2.4.1 of the DisplayPort specification for more information about how to handle each type of reply. ACK: Transaction successful NACK: Transaction failed or is incomplete DEFER: Try again later. I2CNACK: I2C transaction failed or is incomplete I2CDEFER: Try again later. 0 = ACK 1 = NACK 2 = DEFER 4 = I2CNACK 8 = I2CDEFER
11	RW	NOT_PENDING	NO_STOP_ERROR: If the AUX reply from the Sink device did properly terminate with a STOP signal (if the AUX line goes idle), this error bit will be set. There may have been some problem with the connection. The results of the most recent request cannot be guaranteed correct. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
10	RW	NOT_PENDING	SINKSTAT_ERROR: After an HPD IRQ, the hardware will automatically read DPCD addresses 0x205-0x200 and place the results in the DPAUX_DP_AUX_SINKSTAT registers. REPLYTYPE is not updated on this type of read, so if the Sink device does not respond with an ACK, this bit will be set. RX_ERROR, TIMEOUT, and NO_STOP can still be set by the automatic read. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
9	RW	NOT_PENDING	RX_ERROR: This error reporting bit is set if there is an undefined error detected by the AUX channel. Currently, this is only set if the logic was expecting the sink to send a length value (M), but one was not received. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
8	RW	NOT_PENDING	TIMEOUT_ERROR: This error reporting bit is set to PENDING when a timeout has occurred while waiting for a reply. The AUX channel will wait for 400 µs before aborting the AUX transaction. The timeout value is programmable in the DPAUX_DP_AUX_CONFIG register. This bit will remain set until 1 is written to it. 0 = NOT_PENDING 1 = PENDING
7:0	RO	X	REPLY_M: These bits show the number of data bytes an AUX channel transaction receives. An AUX transaction can read or write 16 bytes maximum.

27.11.15 DPAUX_DP_AUX_SINKSTATLO_0

All bits of the SINKSTATLO and SINKSTATHI registers are combined into a 6-byte data buffer. This buffer contains sink/link status information from AUX address space from 00200h to 00205h, as specified in the DisplayPort specification. Hardware automatically issues an AUX read to the above address after a HPD IRQ is detected. The 6 bytes of read data are put into this buffer. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x35..0x38 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	REG

27.11.16 DPAUX_DP_AUX_SINKSTATHI_0

All bits of the SINKSTATLO and SINKSTATHI registers are combined into a 6-byte data buffer. This buffer contains sink/link status information from AUX address space from 00200h to 00205h, as specified in the DisplayPort specification. Hardware automatically issues an AUX read to the above address after a HPD IRQ is detected. The 6 bytes of read data are put into this buffer. The buffer is little endian.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x39..0x3c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	REG

27.11.17 DPAUX_HPD_CONFIG_0

HPD_CONFIG

This register is used to configure the behavior of the HPD plug/unplug events. The INIT values are defined by the DisplayPort specification.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x3d..0x40 | Read/Write: R/W | Reset: 0x07d000fa (0b00000111110100000000000011111010)

Bit	Reset	Description
31:16	0x7d0	UNPLUG_MIN_TIME: The HPD line must be low for this long before it is considered an unplug event. The units of this field are in microseconds.
15:0	0xfa	PLUG_MIN_TIME: The HPD line must be high for this long to be considered a plug event. The units of this field are in microseconds.

27.11.18 DPAUX_HPD_IRQ_CONFIG_0

HPD_IRQ_CONFIG

This register controls the size of the HPD IRQ pulse. The INIT value is defined by the DisplayPort specification.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x41..0x44 | Read/Write: R/W | Reset: 0x000000fa (0bxxxxxxxxxxxxxxxx0000000011111010)

Bit	Reset	Description
15:0	0xfa	MIN_LOW_TIME: IRQ minimum time. The HPD line must be low for at least this amount of time to be an IRQ. An IRQ will occur if IRQ_MIN_TIME < duration of HPD low pulse < UNPLUG_MIN_TIME.

27.11.19 DPAUX_DP_AUX_CONFIG_0

AUX_CONFIG contains miscellaneous configuration parameters for the AUX channel.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x45..0x48 | Read/Write: R/W | Reset: 0x00000190 (0bxxxxxxxxxxxxxxxx0000000110010000)

Bit	Reset	Description
15:0	0x190	TIMEOUT: Sets the wait time, in microseconds, before an AUX transaction will abort and report TIMEOUT in DPAUX_DP_AUXSTAT_TIMEOUT. The default value of 400 µs is specified in the DisplayPort specification.

27.11.20 DPAUX_HYBRID_PADCTL_0

HYBRID_PADCTL

Configuration for the hybrid pads that can be used as AUX/I2C.

This is an array of 1 identical register entries; the register fields below apply to each entry.

Offset: 0x49..0x4c | Read/Write: R/W | Reset: 0x00002462 (0bxxxxxxxxxxxxxxxx0010x10001100010)

Bit	Reset	Description
15	DISABLE	I2C_SDA_INPUT_RCV: Active high receiver enable for the I2C pad's DATA channel. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	DISABLE	I2C_SCL_INPUT_RCV: Active high receiver enable for the I2C pad's CLK channel. 0 = DISABLE 1 = ENABLE
13:12	V0_70	AUX_CMH: Active high 1.0V signal to control the output common mode voltage. 0_60V: VCM = 0.6V (default) 0_64V: VCM = 0.64V 0_70V: VCM = 0.70V 0_56V: VCM = 0.56V 0 = V0_60 1 = V0_64 2 = V0_70 3 = V0_56
10:8	OHM_50	AUX_DRVZ: Active high driver output impedance control. Increasing this value decreases the driver impedance. 0 = OHM_78 1 = OHM_60 2 = OHM_54 3 = OHM_45 4 = OHM_50 5 = OHM_42 6 = OHM_39 7 = OHM_34
7:2	0x18	AUX_DRVI: Active high output driver current control. Increasing the register value increases drive current.
1	ENABLE	AUX_INPUT_RCV: Active high receiver enable for the AUX CH pad. 0 = DISABLE 1 = ENABLE
0	AUX	MODE: Controls whether the pad is in I2C or AUX mode. 0 = AUX 1 = I2C

27.11.21 DPAUX_HYBRID_SPARE_0

PAD_PWR: When a HotPlug is detected on a given DP port, this bit must be set to _POWERUP before any AUX transactions are done. When an Unplug event is detected, this can be written to _POWERDOWN. As long as a DP monitor is connected, the AUX pads should remain powered on because the monitor may send an HPD IRQ, which will require the HW to initiate an AUX read without SW intervention. The logic assumes that the AUX pad will be powered on.

Offset: 0x4d..0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
0	POWERUP	PAD_PWR: Controls the E_PWRD port of the hybrid pad. The pads can consume a lot of power if enabled and should be disabled when not in use. POWERUP: pad has power and can be used for transactions. POWERDOWN: pad is powered down and cannot be used. 0 = POWERUP 1 = POWERDOWN

CHAPTER 28: HDMI CEC

The HDMI Consumer Electronics Control (CEC) block supports CEC standard communication over an HDMI connection. It supports both remote control of HDMI devices attached to the Tegra® X1 device, and also allows other HDMI devices to control Tegra functions. Refer to the CEC appendix of the HDMI specification for details of this link.

28.1 Functional Description

The CEC module is an APB slave. It is located at address NV_ADDRESS_MAP_APB_CEC_BASE (0x70015000).

The CEC consists of hardware state machines that send and receive bytes over the CEC wire. Its simple host interface allows one byte to be sent and received at a time.

The CEC has interrupts for interrupt-driven input and output, and also permits polling I/O.

28.2 Programming Guidelines

28.2.1 Initialization

28.2.1.1 Clock and Reset

The CEC uses a fixed clock source – the APB bus clock. It also has a clock enable and reset in the CAR block.

- CAR.RST_DEVICES_W.SWR_CEC_RST = DISABLE
- CAR.CLK_OUT_ENB_W.CLK_ENB_CEC = ENABLE

28.2.1.2 Interrupt

The CEC is on PRI interrupt controller bit 3.

CEC INT_MASK should be enabled for TX_* and RX_* interrupts. TX_REGISTER_EMPTY and RX_REGISTER_FULL are the key interrupts; most others are errors.

Refer to Chapter 3: Interrupt Controller for more information.

28.2.1.3 Pin Mux

The CEC is available on the HDMI_CEC pin. HDMI_CEC pin mux should be configured for:

- PM=0 (primary CEC function)
- PUPD=NORMAL
- TRISTATE=NORMAL
- E_INPUT=ENABLE
- OD=ENABLE.

28.2.2 CEC Timing

Timing parameters are in units of 32 microseconds. See the HDMI specification for CEC timing requirements. In Tegra X1 devices, the registers have default (reset) values as indicated in the following table.

Register	Field	Value	Comment
RX_TIMING_0	RX_START_BIT_MAX_LO_TIME	0x7a	
	RX_START_BIT_MIN_LO_TIME	0x6d	

Register	Field	Value	Comment
	RX_START_BIT_MAX_DURATION	0x93	
	RX_START_BIT_MIN_DURATION	0x86	
RX_TIMING_1	RX_DATA_BIT_MAX_LO_TIME	0x35	
	RX_DATA_BIT_SAMPLE_TIME	0x21	
	RX_DATA_BIT_MAX_DURATION	0x56	
	RX_DATA_BIT_MIN_DURATION	0x40	
RX_TIMING_2	RX_END_OF_BLOCK_TIME	0x50	
TX_TIMING_0	TX_START_BIT_LO_TIME	0x74	
	TX_START_BIT_DURATION	0x8d	
	TX_BUS_XITION_TIME	0x8	
	TX_BUS_ERROR_LO_TIME	0x71	
TX_TIMING_1	TX_LO_DATA_BIT_LO_TIME	0x2f	
	TX_HI_DATA_BIT_LO_TIME	0x13	
	TX_DATA_BIT_DURATION	0x4b	
	TX_ACK_NAK_BIT_SAMPLE_TIME	0x21	
TX_TIMING_2	BUS_IDLE_TIME_ADDITIONAL_FRAME	0x7	
	BUS_IDLE_TIME_NEW_FRAME	0x5	
	BUS_IDLE_TIME_RETRY_FRAME	0x3	

28.2.3 CEC Control

The CEC has several modes but only some are simulated or supported. TX_RX_MODE=ENABLE should be set last.

Register	Field	Value	Comment
SW_CONTROL	MODE	DISABLE	Use HW mode
INPUT_FILTER	FIFO_LENGTH	0	
	MODE	DISABLE	
HW_CONTROL	RX_LOGICAL_ADDRS	Slave addresses (bitmap)	If slave, holds addresses
	RX_SNOOP	DISABLE	
	RX_NAK_MODE	BLOCK	
	TX_NAK_MODE	BLOCK	
	FAST_SIM_MODE	DISABLE	SW should use DISABLE
	TX_RX_MODE	ENABLE	Enable last
INT_MASK	TX_*, RX_*	DISABLE/ENABLE	ENABLE to enable interrupt

28.2.4 Transmission

Each message is a series of bytes, and each byte has several flags associated with it.

Register	Field	Value	Comment
TX_REGISTER	DATA	byte	8-bit byte
	EOM	1 on last byte	End of message flag
	ADDRESS_MODE	DIRECT/BROADCAST	
	GENERATE_START_BIT	1	
	RETRY_FRAME	0/1	

In general, the first byte contains source and destination addresses, per the CEC specification. The source address is chosen by software as the Tegra address. The destination address is either 15 for broadcast, or less than 15 for a particular slave. The last byte contains EOM=1. When sending to address 15 (broadcast), ADDRESS_MODE must be BROADCAST so that ACK and NAK have correct polarity. If this transmission is a retry of a previous one, RETRY_FRAME should be 1 so that correct retry timing is used.

The TX register interface is unusual in that the interrupt (TX_REGISTER_EMPTY) controls transmission – it's not merely a status signal. To transmit a byte, TX_REGISTER is first written with data, then the INT_STAT.TX_REGISTER_EMPTY interrupt must be cleared. If interrupt service routines are used, they must be careful when clearing an interrupt, because doing so incorrectly could cause a spurious or lost byte.

Transmission pseudo-code:

```
Transmit(unsigned char data, bool eom?, bool broadcast?):
// enqueue byte
Write TX_REGISTER: DATA = data,
        EOM = eom?,
        ADDRESS_MODE = broadcast?,
        GENERATE_START_BIT = 1
// clear interrupt to enable transmission
Write INT_STAT: TX_REGISTER_EMPTY = 1
// wait for transmission
Either Poll INT_STAT.TX_* to go high, or wait for TX_* interrupt.
If TX_REGISTER_EMPTY=1, transfer is complete.
If other TX_* interrupts set, then error.
```

Transmission of multi-block frames requires the blocks (bytes) to be sent back-to-back with no idle space between. If software does not provide the bytes in time, hardware will signal a TX_REGISTER_UNDERRUN interrupt and halt transmission (see Section 28.2.6: Error Recovery below).

28.2.5 Reception

The RX_REGISTER_FULL interrupt indicates that a byte has been received and blocks further reception until cleared. If a subsequent byte is received while RX_REGISTER_FULL is high, an RX_REGISTER_OVERRUN error results.

When a byte is received, two additional EOM and ACK flags are provided as well.

Register	Field	Value	Comment
RX_REGISTER	DATA	byte	8-bit byte
	EOM	1 on last byte	End of message flag
	ACK	bit from bus	Expected value depends on whether broadcast or direct

Once INT_STAT indicates a byte has been received, first read the byte (and flags), and then clear the interrupt.

Reception pseudo-code:

```
Receive(unsigned char &data, bool &eom, bool &ack):
// wait for data
Either Poll INT_STAT.RX_REGISTER_FULL, or
        wait for CEC RX_REGISTER_FULL interrupt
If INT_STAT.RX_* error interrupts set, then error
data, eom, ack = Read RX_REGISTER.DATA/EOM/ACK
// clear interrupt to allow further reception
Write INT_STAT.RX_REGISTER_FULL = 1
```

28.2.6 Error Recovery

In case of transmission error such as UNDERRUN or NAKD, the transmission will stop. Alternatively, the soft reset sequence should be used:

```
tmp = CEC_HW_CONTROL
CEC_HW_CONTROL = 0
CEC_INT_STATUS = 0xffffffff
CEC_HW_CONTROL = tmp
```

28.3 CEC Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

For more information on Consumer Electronics Control (CEC), refer to the CEC appendix of the HDMI specification.

28.3.1 CEC_SW_CONTROL_0

SW controlled mode is not supported. Leave disabled.

Offset: 0x0 | Read/Write: R/W | Reset: 0x000000XX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DISABLE	MODE: 0 = DISABLE 1 = ENABLE
4	RO	X	FILTERED_RX_DATA_PIN
0	RO	X	RAW_INPUT_DATA_PIN

28.3.2 CEC_HW_CONTROL_0

The following steps are the best way to reset the CEC engine:

1. Set the CEC_HW_CONTROL_TX_RX_MODE to DISABLE
2. Set all the interrupt enable bits in CEC_HW_INTR_EN to DISABLE
3. Wait 1 second (1 second is the maximum CEC bus timeout period from the HDMI specification)
4. Set the CEC_HW_CONTROL_TX_RX_MODE to ENABLE and restart.

Other devices on the bus might have been communicating just fine. When the restart has completed, there might be a few spurious false start bits and other receive bus anomalies before normal operation is resumed.

The CEC_HW_CONTROL register contains the control bits and fields for operating and configuring the HW CEC engine.

> ***Note:*** *Once the block is enabled, software should not attempt to change these configuration parameters without first disabling the block, except for the RX_LOGICL_ADDRS and RX_SNOOP fields.*

RX_LOGICAL_ADDRS

The HDMI specification permits a single physical entity to claim logical addresses reserved for different functions (e.g., "Tuner x" and "Playback device y"). For a PC, it is entirely possible to be a multi-function device, so hardware might need to respond to 2 (or more) logical addresses (as many combinations as make sense). The next fields are used to configure the logical addresses that hardware will respond to. As per the specification, hardware will always respond to the broadcast address. This is a 15-bit field, where each bit position is associated with one of the logical addresses. A '1' in any bit position causes the hardware to respond to the associated logical address (i.e., capture data blocks/frames addressed to the address and properly

ack/nak said blocks). To keep things simple, the mapping of bit position to logical address is direct, i.e., bit N maps to logical address N.

RX_SNOOP

When enabled, the hardware will intercept and forward to software all traffic on the CEC bus. It will continue to ack/nak only those addresses that are assigned to it.

RX_NAK_MODE

The HDMI specification is a bit unclear about what the initiator is supposed to do if an intermediate block of a frame is NAK'd, i.e., should the initiator cease transmitting immediately or complete the rest of the frame. The HDMI specification says that a follower may NAK a frame at any time, but the concern is that some initiator might not recover from such an early NAK'd frame. This bit will permit the receive state machine to operate in either mode. The choices are:

- BLOCK which means that a frame will be NAK'd as soon as a RX_REGISTER_OVERRUN is detected
- FRAME which means that the hardware will keep track of any RX_REGISTER_OVERRUNs that occur during frame reception, and NAK only during the last block.

TX_NAK_MODE

The HDMI specification is a bit unclear about what the initiator is supposed to do if an intermediate block of a frame is NAK'd, i.e., should the initiator cease transmitting immediately or complete the rest of the frame. This bit will permit the transmit state machine to operate in either mode. The choices are:

- BLOCK which means transmission will cease at the first NAK'd block
- FRAME which means that transmission will always continue to the end of the frame.

TX_RX_INTERRUPT_ROUTING

The engine generates a single composite interrupt output signal which can then be routed to either PMU or HOST. This bit controls that routing.

TX_RX_MODE

This bit enables or disables the operation of the HW CEC engine. If the TX_RX_MODE is changed to DISABLE, then the HW engine returns to the IDLE state irrespective of what it is doing and drives a 1 onto the CEC bus. If CEC_SW_CONTROL_MODE is ENABLED, the HW CEC engine operation is automatically disabled.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxx0xxxxxxx00000000000000000)

Bit	Reset	Description
31	DISABLE	TX_RX_MODE: 0 = DISABLE 1 = ENABLE
30	DISABLE	FAST_SIM_MODE: 0 = DISABLE 1 = ENABLE
24	BLOCK	TX_NAK_MODE: 0 = BLOCK 1 = FRAME
16	BLOCK	RX_NAK_MODE: 0 = BLOCK 1 = FRAME
15	DISABLE	RX_SNOOP: 0 = DISABLE 1 = ENABLE
14:0	0x0	RX_LOGICAL_ADDRS: All logical addresses that should be matched. One bit per address.

28.3.3 CEC_INPUT_FILTER_0

The CEC_INPUT_FILTER register is used to configure the HW filtering that is required to deglitch the incoming receive data line. As data arrives into the chip, it is pushed into a 1 bit wide FIFO with a maximum of 64 entries deep. The actual used depth of the FIFO is set using the CEC_INPUT_FILTER_FIFO_LENGTH field. The used length is equal to CEC_INPUT_FILTER_FIFO_LENGTH+1. A datum is pushed into the FIFO once per microsecond tick. The FIFO bits should reset to '1' (the idle condition of the CEC bus).

The filtered output of the FIFO is computed roughly as follows:

```
filterMsk[63:0] = (1 << CEC_INPUT_FILTER_FIFO_LENGTH + 1) - 1;

if (reset || (CEC_INPUT_FILTER_MODE == CEC_INPUT_FILTER_MODE_DISABLE))
    fifo[63:0] = 0xffffffffffffffff;
else if (clock tick time)
    fifo[63:0] = (fifo[63:0] << 1) | rawInputDataPin;

if (reset || (CEC_INPUT_FILTER_MODE == CEC_INPUT_FILTER_MODE_DISABLE))
    filteredRxDataPin = 1;
else if ((clock tick time) && ((fifo & filterMsk) == filterMsk))
    filteredRxDataPin = 1;
else if ((clock tick time) && ((~fifo & filterMsk) == filterMsk))
    filteredRxDataPin = 0;
else
    filteredRxDataPin = filteredRxDataPin;
```

This logic should make sure that the filteredRxDataPin output only transitions to 0 or 1 when all the examined bits in the FIFO are also 0 or 1. Thus the filtered data line will not transition as long as there are glitches in the FIFO. The length of 64 provides a maximum operational length of 64 µs.

Every state transition on the filtered receive data line is reportable to software via the interrupt register described later.

Offset: 0x8 | Read/Write: R/W | Reset: 0x000000XX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	DISABLE	MODE: 0 = DISABLE 1 = ENABLE
5:0	X	FIFO_LENGTH

28.3.4 CEC_SPARE_0

Spare register for future use.

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	SPARES

28.3.5 CEC_TX_REGISTER_0

CEC_HW_TX_REGISTER register

This register is used by software to provide the hardware with the actual data to be transmitted on the bus. Note that hardware will 'blindly' transmit what it is given. For example, it will not check to be sure that a proper legal initiator address has been

provided, it will not check to be sure that the maximum frame length of 16 is not violated, etc. These higher level protocol checks are the domain of the SW. In order to facilitate easier software programming and smoother operation, hardware will make its own working copy of the fields in this register, thus quickly freeing up the register for software to write the next block. (Basically, it is a one-deep FIFO with fullness reported via the TX_REGISTER_EMPTY bit).

DATA

The actual 8 bits of address/data to be transmitted on the bus. The data is always transmitted MSB (bit 7) first.

EOM

This is the "end of message" bit for this block of data. This bit is set at the last block of the frame.

ADDRESS_MODE

This bit indicates to the hardware whether this particular block is directly addressed or broadcast addressed (which in turn dictates how hardware is to interpret the ACK/NAK for the block).

GENERATE_START_BIT

Indicates to the hardware that it should precede the transmission of this block of data with a start bit.

RETRY_FRAME

Indicates if the current frame is a retry frame or not. Based on this value, hardware chooses an appropriate bus idle time programmed in TX_TIMING2 register and waits for the bus to be idle before it can attempt to send a start bit. This bit is only meaningful when TX_GENERATE_START_BIT is set.

RETRY FRAME	LAST_FRAM_ SENT_BY_US	WAIT TIME
1	YES	TIMING2_BUS_IDLE_TIME_RETRY_FRAME
1	NO	TIMING2_BUS_IDLE_TIME_RETRY_FRAME
0	NO	TIMING2_BUS_IDLE_TIME_NEW_FRAME
0	YES	TIMING2_BUS_IDLE_TIME_ADDITIONAL_FRAME

If a frame is being sent immediately following the previous frame, hardware waits for "TIMING2_BUS_IDLE_TIME_ADDITIONAL_FRAME". However, if someone else uses the bus before that time elapses, the hardware resets its wait time counter and waits for "TIMING2_BUS_IDLE_TIME_NEW_FRAME".

Offset: 0x10 | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxxxxxxx01xxx0xxx000000000)

Bit	Reset	Description
17	DISABLE	RETRY_FRAME: 0 = DISABLE 1 = ENABLE
16	ENABLE	GENERATE_START_BIT: 0 = DISABLE 1 = ENABLE
12	DIRECT	ADDRESS_MODE: 0 = DIRECT 1 = BROADCAST
8	0x0	EOM
7:0	0x0	DATA

28.3.6 CEC_RX_REGISTER_0

CEC_HW_RX_REGISTER register

This register is used by hardware to buffer data to the software. When a block of data has been received from the bus, it is stored here for software (and the RX_REGISTER_FULL bit is set). The hardware also has a 'working copy' of the receive register to give software as much time as possible to empty it. When a full block is assembled, hardware places it into RX_REGISTER, so the register is like a 1 deep FIFO.

DATA

The 8 bits of data that were read from the bus.

EOM

The EOM bit read from the bus.

ACK_NAK

The ACK/NAK bit as read from the bus. In some cases (broadcast block) even though the device may have ACK'd the frame, some other device on the bus may NAK the frame, and software would need to know this.

Offset: 0x14 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	ACK
8	X	EOM
7:0	X	DATA

28.3.7 CEC_RX_TIMING_0_0

CEC_HW_RX_TIMING0 register

Note: *All timing registers must be configured before the HW CEC engine is enabled.*

RX_START_BIT_MAX_LO_TIME

nominal 3.9 ms, (3900/32 = 122 intervals) = 3.904 ms

The maximum time the received bus can remain low, and still be considered the beginning of a proper start bit. If the start bit low time overruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MIN_LO_TIME

nominal 3.5 ms, (3500/32 = 109 intervals) = 3.488 ms

The minimum time the received bus can remain low and still be considered the beginning of a proper start bit. If the start bit low time underruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MAX_DURATION

nominal 4.7 ms, (4700/32 = 147 intervals) = 4.704 ms

The maximum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration overruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MIN_DURATION

nominal 4.3 ms (4300/32 = 134 intervals) = 4.288 ms

The minimum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration underruns this time, the start bit is considered invalid and is ignored.

Offset: 0x18 | Read/Write: R/W | Reset: 0x86936d7a (0b10000110100100110110110101111010)

Bit	Reset	Description
31:24	0x86	RX_START_BIT_MIN_DURATION
23:16	0x93	RX_START_BIT_MAX_DURATION
15:8	0x6d	RX_START_BIT_MIN_LO_TIME
7:0	0x7a	RX_START_BIT_MAX_LO_TIME

28.3.8 CEC_RX_TIMING_1_0

CEC_HW_RX_TIMING1 register

> ***Note:*** *All timing registers must be configured before the HW CEC engine is enabled.*

RX_DATA_BIT_MAX_LO_TIME

nominal 1.7 ms, (1700/32 = 53 intervals) = 1.696 ms

The maximum time the received bus can remain low and still be considered the beginning of a proper start bit. If the start bit low time overruns this time, the start bit is considered invalid and is ignored.

RX_DATA_BIT_SAMPLE_TIME

nominal 1.05 ms, (1050/32 = 33 intervals) = 1.056 ms

The time to sample that data bit.

RX_DATA_BIT_MAX_DURATION

nominal 2.75 ms, (2750/32 = 86 intervals) = 2.752 ms

The maximum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration overruns this time, the start bit is considered invalid and is ignored.

RX_DATA_BIT_MIN_DURATION

nominal 2.05 ms (2050/32 = 64 intervals) = 2.048 ms

The minimum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration underruns this time, the start bit is considered invalid and is ignored.

Offset: 0x1c | Read/Write: R/W | Reset: 0x40562135 (0b01000000010101100010000100110101)

Bit	Reset	Description
31:24	0x40	RX_DATA_BIT_MIN_DURATION
23:16	0x56	RX_DATA_BIT_MAX_DURATION
15:8	0x21	RX_DATA_BIT_SAMPLE_TIME
7:0	0x35	RX_DATA_BIT_MAX_LO_TIME

28.3.9 CEC_RX_TIMING_2_0

CEC_HW_RX_TIMING2 register

RX_END_OF_BLOCK_TIME

> ***Note:*** *All timing registers must be configured before the HW CEC engine is enabled.*

nominal 1.9 ms, (1900/32 = 80 intervals) = 1.9 ms

The time to wait after the start of the final ACK/NAK phase of frame transmission before returning to the idle state. (Needed since the last ACK/NAK bit will not be followed by another high-to-low transition.)

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000050 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01010000)

Bit	Reset	Description
7:0	0x50	RX_END_OF_BLOCK_TIME

28.3.10 CEC_TX_TIMING_0_0

The next set of timing registers configures the CEC logic for HW assisted operation. To permit flexible configuration (to support possible semi-compliant devices), the bit timing control and check values are programmable (rather than just directly using the specification values).

While the block itself works on a 1 µs tick, in order to save register space, these timing numbers are specified in units of 32 µs.

First, the registers for the various timing intervals (there are quite a few of them).

CEC_HW_TX_TIMING0 register

> ***Note:*** *All timing registers must be configured before the HW CEC engine is enabled.*

For details of exact values and tolerances, refer to the CEC appendix of the HDMI specification.

TX_START_BIT_LO_TIME

nominal 3.7 ms, (3700/32 = 116 intervals) = 3.712 ms

How long to hold the bus lo during the start bit.

TX_START_BIT_DURATION

nominal 4.5 ms, (4500/32 = 141 intervals) = 4.512 ms

The total duration of the start bit

TX_BUS_XITION_TIME

nominal 250 µs, (250/32 = 8 intervals) = 256 µs

Time to wait for bus to settle after transitioning output.

TX_BUS_ERROR_LO_TIME

nominal 3.6 ms (3600/32 = 113 intervals) = 3.616 ms

Time to drive bus low when bus error must be signaled on the bus

Offset: 0x24 | Read/Write: R/W | Reset: 0x71088d74 (0b01110001xxxx10001000110101110100)

Bit	Reset	Description
31:24	0x71	TX_BUS_ERROR_LO_TIME
19:16	0x8	TX_BUS_XITION_TIME
15:8	0x8d	TX_START_BIT_DURATION
7:0	0x74	TX_START_BIT_LO_TIME

28.3.11 CEC_TX_TIMING_1_0

CEC_HW_TX_TIMING1 register

> ***Note:*** *All timing registers must be configured before the HW CEC engine is enabled.*

TX_LO_DATA_BIT_LO_TIME

nominal 1.5 ms, (1500/32 = 47 intervals) = 1.504 ms

How long to hold the bus low when transmitting a '0'.

TX_HI_DATA_BIT_LO_TIME

nominal 0.6 ms, (600/32 = 19 intervals) = 6.08 ms

How long to hold the bus low when transmitting a '1'.

TX_DATA_BIT_DURATION

nominal 2.4 ms, (2400/32 = 75 intervals) = 2.4 ms

The total duration of a data or ACK/NAK bit.

TX_ACK_NAK_BIT_SAMPLE_TIME

nominal 1.05 ms, (1050/32 = 33 intervals) = 1.056 ms

The time to sample that ACK/NAK bit.

Offset: 0x28 | Read/Write: R/W | Reset: 0x214b132f (0b00100001010010110001001100101111)

Bit	Reset	Description
31:24	0x21	TX_ACK_NAK_BIT_SAMPLE_TIME
23:16	0x4b	TX_DATA_BIT_DURATION
15:8	0x13	TX_HI_DATA_BIT_LO_TIME
7:0	0x2f	TX_LO_DATA_BIT_LO_TIME

28.3.12 CEC_TX_TIMING_2_0

CEC_HW_TX_TIMING2 register

This register contains the bus idle time values for the various scenarios indicated in the HDMI specification (Section 9.1). Software indicates to the hardware the amount of delay in units of data bit periods, as defined by the CEC_HW_TX_TIMING1_TX_DATA_BIT_DURATION field.

ADDITIONAL_FRAME:

The amount of time the bus needs to be idle before a new frame can be sent immediately after sending a frame. The suggested value is greater than or equal to 7.

NEW_FRAME:

The amount of time the bus needs to be idle before a new frame can be sent if we were not the initiator for the previous frame. The suggested value is greater than or equal to 5.

RETRY_FRAME:

The amount of time the bus needs to be idle before the same frame can be resent. The suggested value is >= 3.

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000357 (0bxxxxxxxxxxxxxxxxxxxxxx001101010111)

Bit	Reset	Description
11:8	0x3	BUS_IDLE_TIME_RETRY_FRAME
7:4	0x5	BUS_IDLE_TIME_NEW_FRAME
3:0	0x7	BUS_IDLE_TIME_ADDITIONAL_FRAME

28.3.13 CEC_INT_STAT_0

CEC_HW_INTR register

This register contains the individual interrupt and status bits for the CEC HW unit. The CEC_HW_INTR_EN register contains the corresponding enable bit for each interrupt/status bit in the interrupt register. The interrupt enable determines whether the interrupt propagates to the PMIC as an interrupt, but the interrupt status bit itself is always set if the described condition occurs. Software writes a '1' to any interrupt/status bit in order to clear it.

Because the HW state machine also looks at the state of the interrupt bits when determining what action(s) to take, it is important for software to operate in roughly the following order when processing interrupts from this source:

1. Read the interrupt register to determine what caused the interrupt and what needs to be done.
2. Perform the operations required, e.g., reload TX register, read RX register.
3. Finally, clear the corresponding interrupt bits.

TX_REGISTER_EMPTY

The transmit register is empty, so software is free to write the next block of the frame into the register. For multi-block frames, software must in fact provide the next block before the current block is sent, otherwise a TX_REGISTER_UNDERRUN error will occur. When the TX engines reads from the TX_REGISTER register and stores a local copy, this interrupt is asserted.

TX_REGISTER_UNDERRUN

The transmit register was empty at the time the transmit hardware needed to have the next block ready to send. This is an error condition. The transmitter will have stopped transmitting, and recovery will require resending the entire frame from scratch.

TX_FRAME_OR_BLOCK_NAKD

Hardware sets this bit if any block/frame is NAK'd. If software sees this bit set, it knows it will have to resend the frame later.

TX_ARBITRATION_FAILED

Hardware sets this bit during the address block phase of transmitting a frame, if failed to win arbitration. In this case, transmission will cease, and the HW will flush any pending data in the TX_REGISTER. Software will need to retry the frame from scratch (after the appropriate signal free time, i.e., hardware will not automatically retry).

TX_BUS_ANOMALY_DETECTED

Sometime during block transmission, hardware detected anomalous behavior on the bus (e.g., the bus remained low after transmitter had signaled high). When such an anomaly is detected, the transmitter ceases operation, releases the bus high, and flushes any pending data in the TX_TRANSMIT register.

TX_FRAME_TRANSMITTED

This is asserted at the end of the last data bit, i.e., the ACK bit of the last block is sent. This is just an indication that the last block is sent but does not necessarily mean the transmission was successful. Software should still look for other interrupts to check for any errors.

RX_REGISTER_FULL

The hardware has assembled a complete block from the bus and placed it into the RX_REGISTER. Software should read the block immediately to ensure the RX_REGISTER is empty before hardware has the next block ready to load. When the HW RX engine writes RX data to the RX_REGISTER, this bit is set.

RX_REGISTER_OVERRUN

Software failed to read the RX_REGISTER before hardware had another block of data ready to transfer. Hardware will NAK the block/frame, and the initiator will need to retry later. Software will need to flush the partially accumulated frame.

RX_START_BIT_DETECTED

Whenever the receive engine detects a start bit, it will set this bit. This bit is used for debug.

RX_BUS_ANOMALY_DETECTED

The receiver detected some anomaly (including bus errors) on the bus. A bus error has a specific definition in the HDMI specification, but there are other kinds of anomalies that can occur. All anomalies and bus errors are reported here.

RX_BUS_ERROR_DETECTED

The receiver has detected the specific anomaly called a bus error and then signaled a bus error on the bus per the HDMI specification.

FILTERED_RX_DATA_PIN_TRANSITION_H2L

In direct SW bus drive mode, software needs to be interrupted whenever the state of the received data line transitions. This interrupt is asserted on a 1 to 0 transition.

FILTERED_RX_DATA_PIN_TRANSITION_L2H

In direct SW bus drive mode, software needs to be interrupted whenever the state of the received data line transitions. This interrupt is asserted on a 0 to 1 transition

Offset: 0x30 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	FILTERED_RX_DATA_PIN_TRANSITION_L2H: 0 = INACTIVE 1 = ACTIVE
13	X	FILTERED_RX_DATA_PIN_TRANSITION_H2L: 0 = INACTIVE 1 = ACTIVE
12	X	RX_BUS_ERROR_DETECTED: 0 = INACTIVE 1 = ACTIVE
11	X	RX_BUS_ANOMALY_DETECTED: 0 = INACTIVE 1 = ACTIVE
10	X	RX_START_BIT_DETECTED: 0 = INACTIVE 1 = ACTIVE
9	X	RX_REGISTER_OVERRUN: 0 = INACTIVE 1 = ACTIVE
8	X	RX_REGISTER_FULL: 0 = INACTIVE 1 = ACTIVE
5	X	TX_FRAME_TRANSMITTED: 0 = INACTIVE 1 = ACTIVE
4	X	TX_BUS_ANOMALY_DETECTED: 0 = INACTIVE 1 = ACTIVE
3	X	TX_ARBITRATION_FAILED: 0 = INACTIVE 1 = ACTIVE
2	X	TX_FRAME_OR_BLOCK_NAKD: 0 = INACTIVE 1 = ACTIVE
1	X	TX_REGISTER_UNDERRUN: 0 = INACTIVE 1 = ACTIVE
0	X	TX_REGISTER_EMPTY: 0 = INACTIVE 1 = ACTIVE

28.3.14 CEC_INT_MASK_0

Mask register for interrupts. When a field in this register is set to ENABLE, the corresponding field with a value of 1 in the INT_STATUS register will result in assertion of an interrupt line.

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000xx000000)

Bit	Reset	Description
14	DISABLE	FILTERED_RX_DATA_PIN_TRANSITION_L2H: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13	DISABLE	FILTERED_RX_DATA_PIN_TRANSITION_H2L: 0 = DISABLE 1 = ENABLE
12	DISABLE	RX_BUS_ERROR_DETECTED: 0 = DISABLE 1 = ENABLE
11	DISABLE	RX_BUS_ANOMALY_DETECTED: 0 = DISABLE 1 = ENABLE
10	DISABLE	RX_START_BIT_DETECTED: 0 = DISABLE 1 = ENABLE
9	DISABLE	RX_REGISTER_OVERRUN: 0 = DISABLE 1 = ENABLE
8	DISABLE	RX_REGISTER_FULL: 0 = DISABLE 1 = ENABLE
5	DISABLE	TX_FRAME_TRANSMITTED: 0 = DISABLE 1 = ENABLE
4	DISABLE	TX_BUS_ANOMALY_DETECTED: 0 = DISABLE 1 = ENABLE
3	DISABLE	TX_ARBITRATION_FAILED: 0 = DISABLE 1 = ENABLE
2	DISABLE	TX_FRAME_OR_BLOCK_NAKD: 0 = DISABLE 1 = ENABLE
1	DISABLE	TX_REGISTER_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	DISABLE	TX_REGISTER_EMPTY: 0 = DISABLE 1 = ENABLE

28.3.15 CEC_HW_DEBUG_RX_0

Offset: 0x38 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
27	X	RXDATABIT_SAMPLE_TIMER
26	X	LOGICADDR_MATCH
25	X	FORCELOOUT
24:21	X	STATE
20:17	X	RXBIT_COUNT
16:0	X	DURATION_COUNT

28.3.16 CEC_HW_DEBUG_TX_0

Offset: 0x3c | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	TXDATABIT_SAMPLE_TIMER
25	X	FORCELOOUT
24:21	X	STATE
20:17	X	TXBIT_COUNT

Bit	Reset	Description
16:0	X	DURATION_COUNT

28.3.17 CEC_HW_SPARE_0_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH

CHAPTER 29: MIPI-CSI (CAMERA SERIAL INTERFACE)

This chapter describes the Tegra® X1 Camera Serial Interface (CSI), which is based on the MIPI CSI-2 v1.1 standard specification. The MIPI CSI-2 v1.1 standard is available from MIPI to its members at http://www.mipi.org/.

29.1 Functional Description

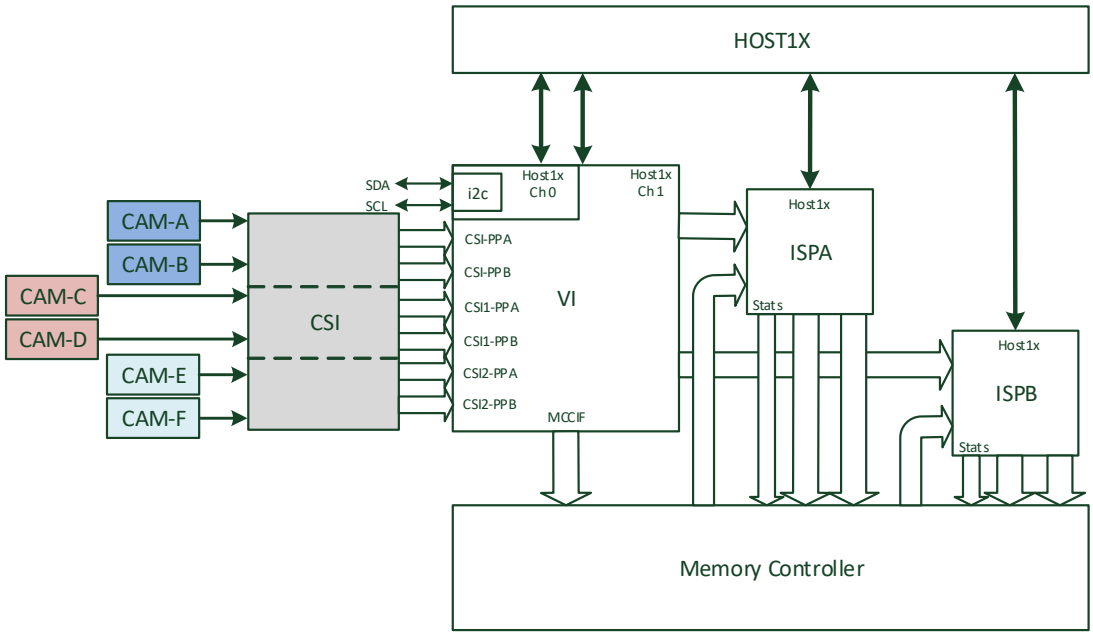
The Tegra X1 CSI unit implements MIPI compliant CSI receivers that may receive data from an external camera module with a CSI transmitter. The CSI unit provides for direct connection to six camera modules (A, B, C, D, E, and F), all of which may be active simultaneously: Internally, the CSI unit is composed of three identical instances referred to as CSI, CSI1, and CSI2 as follows:

- CSI: Supports up to 4 lanes for either Camera A or B with only one camera active, or up to 2 lanes for Cameras A and B when configured in the 2x2 mode with both camera's active. Two clock lanes are available to manage the Camera A and B streams.
- CSI1: Supports up to 4 lanes for either Camera C or D with only one camera active, or up to 2 lanes for Cameras C and D when configured in the 2x2 mode with both camera's active. Two clock lanes are available to manage the Camera C and D streams.
- CSI2: Supports up to 4 lanes for either Camera E or F with only one camera active, or up to 2 lanes for Cameras E and F when configured in the 2x2 mode with both camera's active. Two clock lanes are available to manage the Camera E and F streams.

The CSI unit offers both single and multi-camera configuration options. The single camera configuration option is supported with a 1-to-4 lane sensor connected to one of the CSI, CSI1, or CSI2 interfaces. Several multi-camera configuration options are also available by connecting up to two cameras to one of the CSI, CSI1, or CSI2 interfaces or by operating the interfaces simultaneously with multiple cameras connected. A maximum of six cameras can be supported by the CSI unit if all interfaces are utilized.

The following figure illustrates the datapaths associated with the Tegra X1 camera subsystem.

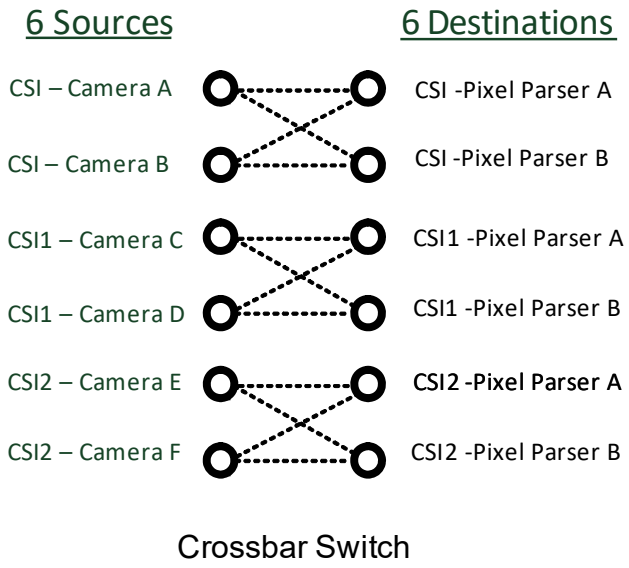
Figure 115: Datapath Through the Tegra X1 Camera Subsystem



A maximum of six data/pixel streams can be processed simultaneously at any given time. The streams can come from any of the six possible sources, as shown below. If the two streams come from a single source, then the streams are separated using

a filter indexed on different virtual channel numbers or data types. In case of separation using data types, the normal data type is separated from the embedded data type. Because there are only two pixel parsers for each CSI, CSI1, and CSI2 instance (PPA and PPB), the virtual channel and embedded data capability cannot be used at the same time.

Figure 116: Data Source/Destination Options



The CSI unit may receive data from camera modules through the MIPI serial interface. A test pattern generator has been implemented to assist validation.

The CSI unit supports single-shot mode and burst capture mode.

The CSI unit is designed with error resilience.

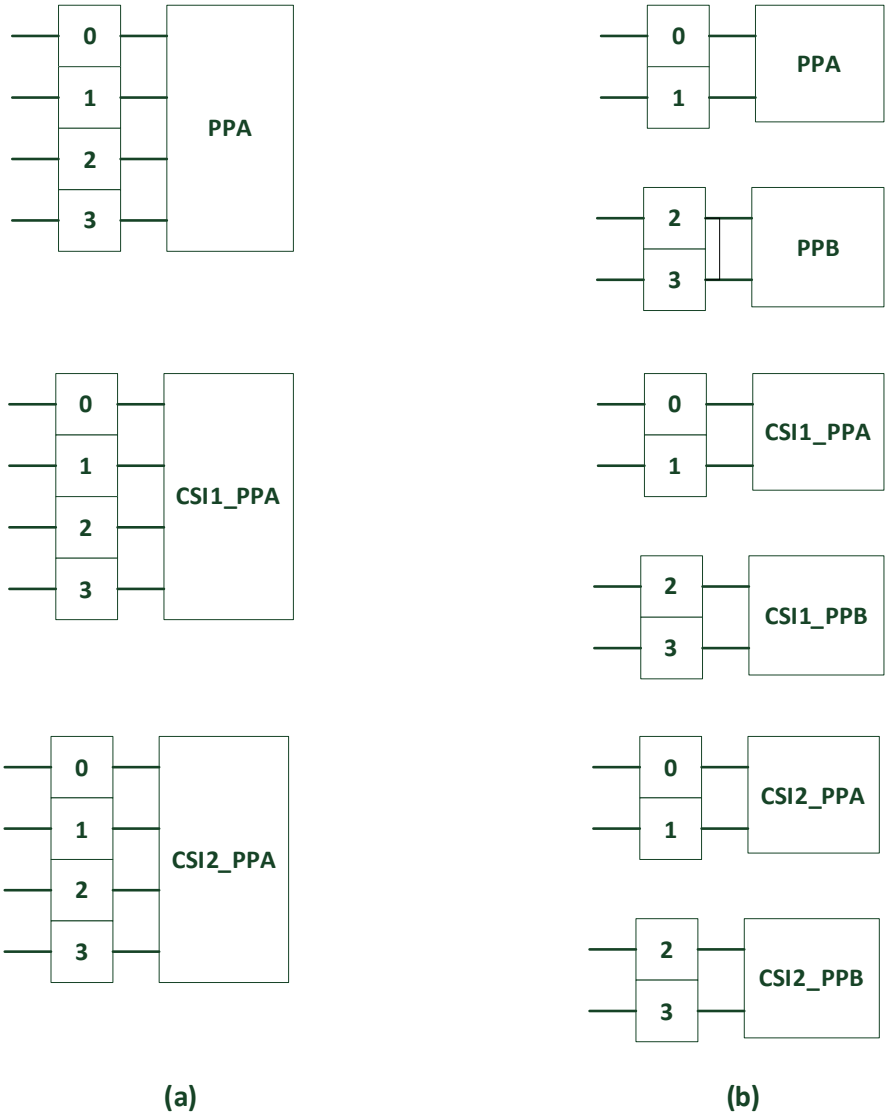
29.1.1 Lane Configurations

The number of MIPI lanes supported provides flexibility in the support of both single and dual camera user models. The typical single and dual camera application targets include high-resolution mono image capture in addition to dual camera image capture configurations:

- 1 x 4 (Single camera with 4 lane sensor using CSI_PPA, CSI1_PPA, or CSI2_PPA)
- 2 x4 (Stereo pair with 4 lanes for each camera using any pair of CSI_PPA, PPA, CSI1_PPA, and CSI2_PPA)
- 2 x 2 (Dual camera mode for CSI, CSI1, and CSI2 supporting 2,4, or 6 camera streams)

The following figure summarizes the typical camera configurations.

Figure 117: CSI MIPI Lane Configurations



Note: *Dual camera streams may be routed to either PPA or PPB when operating in 2x2 mode for each instance (CSI, CSI1, or CSI2) but cannot cross the CSI instance boundaries.*

There are several possible configuration options depending on the system configuration. A summary of the lane configuration options for various 1x4 and 2x4 configurations is provided in the following table.

Table 164: Pin Connections for Lane Configuration Options for 1x4 and 2x4

MIPI Pads		Pin Connects for Lane Configuration Options					
		1x4 (Option A)	1x4 (Option B)	1x4 (Option C)	2x4 (Option A)	2x4 (Option B)	2x4 (Option C)
csiab_pad	CLK	csi_a_clk_n csi_a_clk_p	-	-	-	csi_a_clk_n csi_a_clk_p	csi_a_clk_n csi_a_clk_p
	Data Lanes	csi_a_d0_n csi_a_d0_p	-	-	-	csi_a_d0_n csi_a_d0_p	csi_a_d0_n csi_a_d0_p
		csi_a_d1_n csi_a_d1_p	-	-	-	csi_a_d1_n csi_a_d1_p	csi_a_d1_n csi_a_d1_p
		csi_b_d0_n csi_b_d0_p	-	-	-	csi_b_d0_n csi_b_d0_p	csi_b_d0_n csi_b_d0_p
		csi_b_d1_n csi_b_d1_p	-	-	-	csi_b_d1_n csi_b_d1_p	csi_b_d1_n csi_b_d1_p
	CLK	csi_b_clk_n csi_b_clk_p	-	-	-	csi_b_clk_n csi_b_clk_p	csi_b_clk_n csi_b_clk_p
csicd_pad	CLK	-	csi_c_clk_p csi_c_clk_n	-	csi_c_clk_p csi_c_clk_n	csi_c_clk_p csi_c_clk_n	-
	Data Lanes	-	csi_c_d0_n csi_c_d0_p	-	csi_c_d0_n csi_c_d0_p	csi_c_d0_n csi_c_d0_p	-
		-	csi_c_d1_n csi_c_d1_p	-	csi_c_d1_n csi_c_d1_p	csi_c_d1_n csi_c_d1_p	-
		-	csi_d_d0_n csi_d_d0_p	-	csi_d_d0_n csi_d_d0_p	csi_d_d0_n csi_d_d0_p	-
		-	csi_d_d1_n csi_d_d1_p	-	csi_d_d1_n csi_d_d1_p	csi_d_d1_n csi_d_d1_p	-
		-	csi_d_clk_n csi_d_clk_p	-	csi_d_clk_n csi_d_clk_p	csi_d_clk_n csi_d_clk_p	-
csief_pad	CLK	-	-	csi_e_clk_n csi_e_clk_p	csi_e_clk_n csi_e_clk_p	-	csi_e_clk_n csi_e_clk_p
	Data Lanes	-	-	csi_e_d0_n csi_e_d0_p	csi_e_d0_n csi_e_d0_p	-	csi_e_d0_n csi_e_d0_p
		-	-	csi_e_d1_n csi_e_d1_p	csi_e_d1_n csi_e_d1_p	-	csi_e_d1_n csi_e_d1_p
		-	-	csi_f_d0_n csi_f_d0_p	csi_f_d0_n csi_f_d0_p	-	csi_f_d0_n csi_f_d0_p
		-	-	csi_f_d1_n csi_f_d1_p	csi_f_d1_n csi_f_d1_p	-	csi_f_d1_n csi_f_d1_p
	CLK	-	-	csi_f_clk_n csi_f_clk_p	csi_f_clk_n csi_f_clk_p	-	csi_f_clk_n csi_f_clk_p

Each of the MIPI x4 bricks may also be configured as a 1x2 or as a 2x2, resulting in the ability to configure from 1 to 6 2-lane cameras.

29.2 Use Cases

The use cases supported by the CSI unit include basic single camera, multiple cameras, embedded data, and virtual channel. The still and video capture sequences involve different single camera and multi-camera configuration options.

The table below summarizes the use case categories.

Table 165: Summary of CSI Camera Use Cases

Case	Description	Sources	Destinations
Single Camera Video Capture	Basic preview or video capture using single camera	Camera A, B, C, D, E, or F	PPA, PPB, CSI1_PPA,CSI1_PPB, CSI2_PPA,or CSI2_PPB

Table 165: Summary of CSI Camera Use Cases

Case	Description	Sources	Destinations
Stereo Camera Video Capture	Stereo pair video capture	Camera AB, CD, or EF	PPA and PPB, CSI1_PPA and CSI1_PPB, or CSI2_PPA and CSI2_PPB
Single Camera Still Capture and Preview	Basic high resolution still capture supporting single shot and burst capture and preview	Camera A, B, C, D, E, or F	PPA, PPB, CSI1_PPA,CSI1_PPB, CSI2_PPA,or CSI2_PPB
Stereo Camera Still Capture and Preview	Stereo camera high resolution still capture and preview	Camera AB,CD, or EF	PPA and PPB, CSI1_PPA and CSI1_PPB, or CSI2_PPA and CSI2_PPB
Dual Camera Mono Video Capture	Video feed of front facing camera and user facing camera for video annotation or video preview during Video Conference	Any camera pair	Any pair or pixel parsers
Video Conference Mono Still Capture & Preview*	Video feed of user facing camera for video conference and mono still capture from front facing camera.	Any camera pair	Any pair or pixel parsers
Multi-Camera Automotive	Low resolution YUV or RGB video feed from two cameras.	Any grouping of the 6 cameras	Any grouping of the 6 pixel parsers
Virtual Channel(1)	One stream goes to both parsers.	Camera AB,CD, or EF	PPA and/or PPB CSI1_PPA and CSI1_PPB, or CSI2_PPA and CSI2_PPB
Embedded Data	One stream goes to one parser and has normal and embedded data	Camera AB,CD, or EF	PPA and PPB, CSI1_PPA and CSI1_PPB, or CSI2_PPA and CSI2_PPB
Virtual channel based interleaving is not supported.			

The CSI unit supports both progressive and interlaced video captures. The interlaced capture is targeted toward the automotive use case and will always be either RGB or YCbCr data. During interlaced video capture, the Top and Bottom frames are signaled to the VI2 during the SOF by togging the LSB on the CSI[A,B]2VI_DATA bus. Refer to Chapter 31: Video Input (VI) in this TRM for more information.

29.3 Performance

The following table describes the various MIPI lane configurations and the theoretical maximum Mpixels/s achievable at 100% efficiency for several RAW Bayer pixel precisions and YCbCr 4:2:2.

Table 166: Maximum Capture Rate in MPixels/s for Various CSI Configurations

Link Speed	Number of Lanes	Maximum Mpixels/s with 100% Efficiency				
		RAW8	RAW10	RAW12	RAW14	YCbCr422
600 Mbps	1	75	60	50	42.9	37.5
	2	150	120	100	85.7	75
	3	225	180	150	128.6	112.5
	4	300	240	200	171.4	150
800 Mbps	1	100	80	66.7	57.1	50
	2	200	160	133.3	114.3	100
	3	300	240	200	171.4	150
	4	400	320	266.7	228.6	200
1 Gbps	1	125	100	83.3	71.4	62.5
	2	250	200	166.7	142.9	125
	3	375	300	250	214.3	187.5
	4	500	400	333.3	285.7	250

Table 166: Maximum Capture Rate in MPixels/s for Various CSI Configurations

Link Speed	Number of Lanes	Maximum Mpixels/s with 100% Efficiency				
1.5 Gbps	1	187.5	150	125	107.1	93.8
	2	375	300	250	214.3	187.5
	3	562.5	450	375	321.4	281.3
	4	750	600	500	428.6	375

The maximum throughput supported by the CSI unit using all available lanes is described by the following equation:

Maximum BW in bits/s = (1.5 Gbps/lane)*(12 lanes) = 18 Gbps

For RAW10 (10bpp), a total of 1.8G pixels/s can be delivered to the VI/ISP subsystem; similarly for YCbCr422 (16bpp), a total of 1.125 Gpixels per second can be delivered. Typically, overhead due to blanking and protocol will reduce these numbers by approximately 15%.

The following table translates the maximum pixel rate into the maximum achievable frame rate for different sensor resolutions and bit depths.

Table 167: Maximum Capture Rate in FPS for Various CSI Configurations

Frame Size	Pixel Type	Maximum Frame Capture Rate (FPS) with 100% Efficiency											
		800 Mbps				1 Gbps				1.5 Gbps			
		1	2	3	4	1	2	3	4	1	2	3	4
8MP	RAW10	10	20	30	40	12.5	25	37.5	50	18.8	37.6	56.4	75.2
	RAW12	8.3	16.7	25	33.4	10.4	20.8	31.1	41.6	15.6	31.2	46.8	62.4
	RAW14	7.2	14.3	21.5	28.6	8.9	17.9	26.8	35.7	13.4	26.8	40.2	53.6
14MP	RAW10	5.5	10.9	16.4	21.9	6.8	13.7	20.5	27.3	10.7	21.4	32.1	42.8
	RAW12	4.6	9.1	13.7	18.2	5.7	11.4	17.1	22.8	8.9	17.8	26.7	35.6
	RAW14	3.9	7.8	11.7	15.6	4.9	9.8	14.7	19.5	7.7	15.4	23.1	30.8
16MP	RAW10	4.9	9.9	14.8	19.8	6.2	12.4	18.6	24.7	9.4	18.8	28.2	37.6
	RAW12	4.1	8.2	12.4	16.5	5.2	10.3	15.5	20.6	7.8	15.6	23.4	31.2
	RAW14	3.5	7.1	10.6	14.1	4.4	8.8	13.3	17.7	6.7	13.4	20.1	26.8
24MP	RAW10	3.3	6.6	9.8	13.1	4.1	8.2	12.3	16.4	6.3	12.6	18.9	25.2
	RAW12	2.7	5.5	8.2	10.9	3.41	6.8	10.3	13.7	5.2	10.4	15.6	20.8
	RAW14	2.3	4.7	7	9.4	2.9	5.9	8.8	11.7	4.5	9	13.5	18
32MP	RAW10	2.5	5.1	7.6	10.2	3.2	6.4	9.5	12.7	4.7	9.4	14.1	18.8
	RAW12	2.1	4.2	6.4	8.5	2.7	5.3	8	10.6	3.9	7.8	11.7	15.6
	RAW14	1.8	3.6	5.5	7.3	2.3	4.5	6.8	9.1	3.3	6.6	9.9	13.2

To minimize the rolling shutter artifacts, a generally accepted minimum still capture target of $1/15^{th}$ of a second is desired. The Zero Shutter Lag (ZSL) user model in which a preview stream and a still capture circular buffer are maintained makes it desirable to achieve the highest frame rate possible to provide good motion video for scene composition.

29.4 Supported CSI to VI Data Formats

The formats of the pixel data presented from the CSI to the VI across the 24-bit bus for each of the pixel parsers are summarized in the table below.

The Tegra X1 device presents all Raw data to the VI/ISP in its original bit order.

Table 168: CSI to VI Pixel Formats

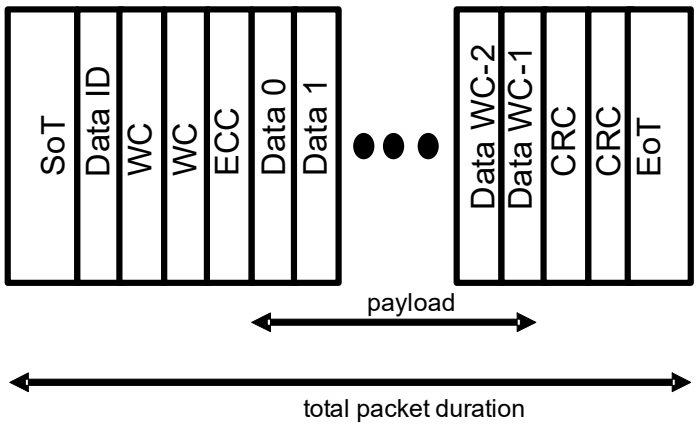
Format Name		23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
6-bit raw or DPCM 12-6-12/ 10-6-10		0[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
7-bit raw or DPCM 12-7-12/ 10-7-10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D6	D5	D4	D3	D2	D1	D0
8-bit raw or DPCM 14-8-14/ 12-8-12/10-8-10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
10-bit raw or DPCM 14-10-14		0	0	0	0	0	0	0	0	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
12-bit raw		0	0	0	0	0	0	0	0	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
14-bit raw		0	0	0	0	0	0	0	0	0	0	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
8-bit arbitrary/ embedded		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
16-bit RGB (RGB565)		B4	B3	B2	B1	B0	0	0	0	G5	G4	G3	G2	G1	G0	0	0	R4	R3	R2	R1	R0	0	0	0
15-bit RGB (RGB555)		B4	B3	B2	B1	B0	0	0	0	G4	G3	G2	G1	G0	0	0	0	R4	R3	R2	R1	R0	0	0	0
24-bit RGB (RGB888)		B7	B6	B5	B4	B3	B2	B1	B0	G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0
12-bit RGB (RGB444)		B3	B2	B1	B0	0	0	0	0	G3	G2	G1	G0	0	0	0	0	R3	R2	R1	R0	0	0	0	0
18-bit RGB (RGB666)		B5	B4	B3	B2	B1	B0	0	0	G5	G4	G3	G2	G1	G0	0	0	R5	R4	R3	R2	R1	R0	0	0
YUV422 8-bit	1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 8-bit (legacy)	Odd Line[2] 1st CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Odd Line 2nd CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 8-bit (CSPS)	Odd Line	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10

1. The bits marked as '0' are actually don't cares.
2. Odd Line refers to the first line from the CSI unit.

29.5 CSI Packet Structure

The CSI packet structure is illustrated below. It has a start-of-transmission sequence (SoT), which contains a 1-byte preamble sequence. The packet is composed of a 4-byte header, a body of word count (WC) bytes, and a 2-byte CRC check. The end-of-transmission (EoT) is indicated by the line state going from high-speed transmission to the LP11 state.

Figure 118: CSI Packet Structure



29.6 CSI Implementation

The CSI interface consists of:

- MIPI D-PHY block
- Control Interface Logic (CIL) for Serial Clock Receiver
- CSI datapath module which is implemented as a sub-module of VI module

In the Tegra X1 implementation, three CSI bricks including the csiab_pad with 4x data lanes, the csicd_pad with 4x data lanes and the csief_pad with 4x data lanes receive serial data from cameras, deserialize them into 8-bit parallel data, and send them to the CSICIL block. The CSICIL block performs packet boundary detection by searching for the packet preamble. The main CSI module receives an 8-bit packet aligned data from CSICIL, performs parity checking on the header, header decoding, CRC check, and pixel parsing. The output is sent out to the VI through the 28-bit bus, which outputs 1 pixel per VI clock.

29.7 Performance Limitations

The performance of the CSI interface is subject to several factors that must all be met. The performance factors that are directly related to the CSI interface are the serial data rate and the internal pixel rate. Maximum serial data rate is expected to be 1.5 Gbps/lane given the ideal system condition. This data rate may be degraded depending on system design and may also be limited by the CSI transmitter in the camera.

For each CSI data lane, the serial data stream received by the CSI PHY is internally converted to a byte stream by the CSI Control Interface Logic (CIL) before further processed by the CSI module that resides as part of the Video Input (VI) module. The CSI byte clock is used to transfer this data byte stream (1 byte per data lane) between the CIL and the CSI datapaths. This CSI byte clock is derived from the received CSI serial clock by dividing the CSI serial clock by 4. So for 1 Gbps serial CSI data rate, the serial clock frequency is 500 MHz and the byte clock frequency is 125 MHz.

The byte stream received by the CSI CIL logic is converted to a pixel stream at 1 pixel per clock in the CSI module. This pixel stream output is further processed by other modules (ISP or the rest of the VI datapath). Depending on where the CSI pixel stream is processed (ISP or VI), the maximum pixel clock frequency is limited by the pixel clock frequency of the modules that process this pixel stream.

Note: *VI/ISP pixel frequency may be lower than CSI output pixel clock frequency and therefore, may limit the actual pixel data rate. Please refer to the ISP and VI specifications.*

With multiple data lanes per CSI interface, the maximum serial data link speed may not be achievable due to pixel clock frequency limitations. Also, if two data streams come from the same CSI interface, since the streams are interleaved in time, the pixel clock frequency limitations may also limit the frame rate of the combined streams.

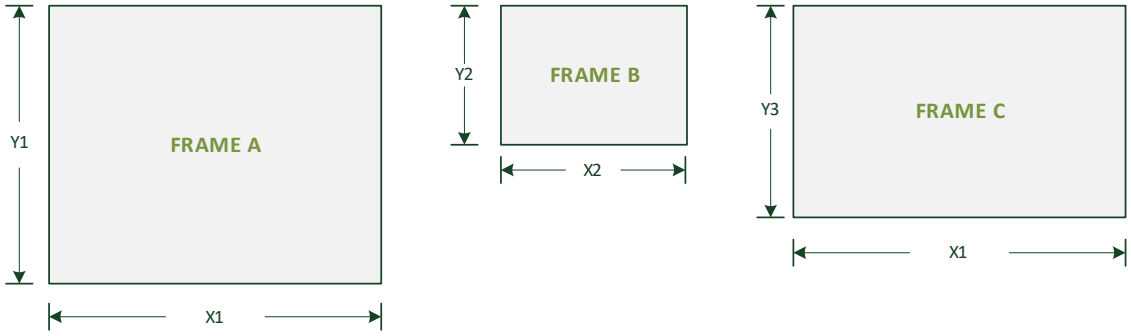
Table 169: Packet Efficiency

	Best* WC=65536	Typical* WC=1024
Payload	524 µs	8192 ns
Overhead	356 ns	356 ns
Efficiency	99.9%	96%

29.8 Frame Size Mismatch Scenarios

Several frame mismatch scenarios can occur during image capture. In these situations, a frame might not have the expected dimensions as shown in the following figure.

Figure 119: Frame Size Variations



Frame A = Full Resolution Image of size (X1,Y1)
Frame B - Preview image of size (X2,Y2) where, Y2 < Y1 & X2 < X1
Frame C - Preview image of size (X1,Y3) where, Y3 < Y1

The following subsections describe the behavior of the CSI unit (and VI/ISP) in the various frame size mismatch scenarios:

29.8.1 Received Frame Width and Height Matches Programmed Width and Height

When the CSI unit receives any frame size, it starts capturing the frame. The VI will raise the sync point when it receives the EoF at the right time.

29.8.2 Received Frame Width Does Not Match Programmed Width

Packet Width Mismatch Error

If the CSI unit receives any frame size in which the width of the incoming frame does not match the programmed value, the error is detected and signaled to the VI. The VI will drop the frame in the case of packet width mismatch errors.

29.8.3 Received Frame Height Does Not Match Programmed Height

CASE 1: Frame Height Mismatch Error - Fewer Rows Than Expected

Example: The CSI unit is instructed to capture Frame A:

When the sensor receives Frame C, it starts capturing the frame as the width of the frame matches the expected frame width. The CSI unit will encounter an early EoF. At this time, it will rearm the pixel parser state-machine with the same set of frame capture parameters and wait for the next frame.

If this frame was to be sent to memory, the VI unit will not detect that it is not the right frame and will not raise the sync point. When it receives the next frame from the CSI unit, it will overwrite the memory buffer (at the buffer location pointed by previous frame parameter).

If this frame was to be sent to the ISP, the VI forwards the EoF control packet to the ISP and indicates that the Frame was too short. The CSI and VI will not pad the frame to make the frame the right size. The ISP unit needs to detect the error condition and should not update the parameter for the next frame and apply the same parameters (as the previous frame) for the next incoming frame from the VI.

CASE 2: Frame Height Mismatch Error - More Rows Than Expected

Example 2: The CSI unit is instructed to capture Frame C:

When the CSI unit receives Frame A, it starts capturing the frame as the width of the frame is matched.

For the frame to be sent to memory, the VI marks the last atom of the frame to the memory as a non-posted write. The VI then raises a sync point when the WrAck and EoF at the expected frame size if received. In this case, instead of EoF, the VI will receive more pixel data. On detecting this condition, the VI will not raise the sync point; it drops the rest of the frame and does not corrupt the memory buffer. The CSI unit will rearm the pixel parser state machine with the same set of frame capture parameters and waits for the next frame. Because the VI is also aware of the expected frame height and detects the frame to be erroneous, it will overwrite the memory buffer with the next frame.

In case this frame was to be sent to the ISP, the VI, when it receives the regular pixel data instead of EoF flag from the CSI unit, will drop the data and generate an EoF flag to the ISP to terminate the frame and mark it as frame-too-big. The ISP unit needs to detect the error condition; it should not update the parameter for the next frame and will apply the same parameters (as the previous frame) for the next incoming frame from the VI.

29.8.4 Double Buffering of the Configuration Registers

There are two copies of configuration registers: operational and the shadow registers. The shadow registers are updated from the Host1x input when it does not have a pending capture command. The shadow register is copied to the operational register when a frame is successfully captured (at the EoF received at the right time). In the event of a frame size mismatch, the shadow registers are not copied and the current settings remain active for the next incoming frame:

1. Successful Frame Capture: The CSI pixel parser (and header parser) sub-units copy the shadow configuration register to the operational configuration register, if it needs to capture the next frame. This configurations need to move along the pipeline with the SoF marker.
2. Failed Frame Capture (Dropped): When a frame is dropped for being too big or too small, the shadow registers are not copied over to the operational register and the CSI remains armed for the next frame.

Whenever the CSI drops an incoming frame, whether due to a mismatch in virtual channel, data type, width, or the height, this information is appropriately logged in the status register. This would help in debugging any system issues where the software arms the CSI to capture the frame but eventually times out due to a missing sync point. There is only one copy of this register (per CSI channel) and every subsequent dropped frame will update this status register.

29.9 Error Handling

The CSI unit is required to tolerate and/or recover stream errors at various levels.

Table 170: CSI Error Categories

Error Categories	Classification	Type	Description
D-PHY Level Errors	Start of Transmission (SoT) Error	Single-bit error	Corrected an error condition signal from PHY/CIL
		Multi-bit error	Packet discarded
	Control Error	Incorrect line state sequence	XError signal asserted
	Escape Entry Error	Escape command different from 8-bits and escape handshaking incorrect	XError signal asserted
	End of Transmission (EoT) Error	A bit of the payload not on a byte boundary	Unrecoverable error condition signaled to the application layer
Packet Level Errors	Packet Header Error	Single bit error	Detected and corrected by the ECC code
		Multi-bit error	Detected but not corrected; error is flagged and interrupt generated
	Packet Payload Errors	Signaled through CRC Code	Detected but not corrected; error is flagged and interrupt generated
Protocol Decoding Errors	Frame Sync Error	FE is not matched to FS	Frame data is supplied to the VI and an error is signaled as a stream error to the application layer.
	Unrecognized ID	Data ID Error	Packet discarded
CSI Block Specific Errors	Packet Header Mismatch Error	Data ID Error	If the virtual channel ID does not match the programmed value for the corresponding pixel parser then the packet is discarded
		Data Type Error	If the data-type does not match the programmed value for the corresponding pixel parser then the packet is discarded
	FIFO Overflow	Backpressure	Frame is dropped
	Packet Word Count Mismatch	Packet longer than word count	Packet is dropped
		Packet shorter than word count	Packet is dropped
	Frame Height Mismatch	Frame larger than programmed size	Frame data is supplied to the VI until EoF and then the pixel parser is re-armed. VI will receive or drop the additional data depending on destination (MEMC or ISP)
		Frame smaller than programmed size	The pixel parser is rearmed on EoF and waits for the next frame

29.10 Other Architectural Constraints

The following specifies other architectural constraints for this design:

- Because there are only 2 pixel parsers for the CSI, CSI1, or CSI2 instance, embedded data and virtual channel cannot be supported simultaneously.
- Embedded data in a frame can be output in the same output port as the pixel data stream; however, in some cases where image processing or data reformatting is performed in VI/ISP, this embedded data may be accidentally processed in VI/ISP. Currently, VI/ISP cannot differentiate between embedded data or pixel data. Refer to the VI chapter for more information on the options for managing embedded data.

29.11 CSI Datapath Module

The CSI, CSI1, and CSI2 instance datapaths consist of two input ports and two output ports. For each instance, there are two datapaths for pixel stream processing; therefore, at any time, a maximum of two input ports may be active simultaneously.

The components of the CSI, CSI1, and CSI2 datapaths include:

- Two input ports for each instance: CSI A interface and CSI B interface, CSI1 A and CSI1 B, and CSI2 A and CSI2 B. Each port is capable of carrying a stream with CSI compatible data format.
- Two asynchronous input buffers (FIFOs) for each instance to receive streams from the CSI, CSI1, and CSI2 A interface and the CSI, CSI1, and CSI2 B interface.
- Two header parsers for CSI, CSI1, and CSI2 for searching packet header and to perform error detection and correction of CSI header.
- Two pixel parsers with corresponding output ports for CSI, CSI1, and CSI2. Each pixel parser can convert a CSI packet data stream to output a pixel stream. Each output port consists of a maximum 24 bits of data per clock. Depending on the output data format options, one or two pixels per clock may be sent.

29.11.1 Header Parser

CSI pixel stream processing mainly consists of header parser and pixel parser. There are 2 header parsers: header parser A and header parser B.

Header parser A is enabled when CSI A interface is selected as input source. Similarly, header parser B is enabled when CSI B interface is selected as input source.

In general the header parser is used for:

- Detecting long and short packet headers and deciding what to do with the packet. This includes performing error detection and correction on the packet header prior to making decision and dealing with uncorrectable packet header.
- Skipping extra data on longer than expected data packets.
- Skipping next packet on imminent input buffer overflow when the pixel parser is busy processing current packet.

When enabled, the header parser will interpret CSI packet header, perform error detection and correction. If the packet header is uncorrectable, then an error is flagged and interrupt generated. If good or correctable CSI packet header is found, then the header parser will check the Data Identifier (DI) byte in the packet header and check the 2-bit Virtual Channel (VC) identifier and the 6-bit Data Type (DT) within this Data Identifier byte and will also check the 2-byte Word Count (WC) value in the packet header. If the virtual channel ID does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the virtual channel ID matches the expected (programmed) value for the corresponding pixel parser then the header parser must decide what to do with the packet depending on the Data Type and the Word Count value. For long packet, if the Data Type does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the Data Type of the long packet matches the expected (programmed) value for the corresponding pixel parser, then the corresponding pixel parser is notified to process the remaining packet data and checksum.

When first enabled, the header parser must always search for Frame Start packet which is a CSI short packet. When frame start packet/signal is found, the header parser will notify the corresponding pixel parser so that it can prepare to process a new frame and output frame start code.

29.11.1.1 Data Type

There are 64 possible data types based on the 6-bit Data Type value.

Table 171: Possible Data Types

Data Type	Description
0x00 – 0x07	Synchronization Short Packet Data Types
0x08 – 0x0F	Generic Short Packet Data Types
0x10 – 0x17	Generic Long Packet Data Types
0x18 – 0x1F	YUV Data
0x20 – 0x27	RGB Data
0x28 – 0x2F	RAW Data
0x30 – 0x37	User Defined Byte-based Data
0x38 – 0x3F	Reserved

29.11.1.2 Short Packets

There are 16 possible short packets based on Data Type value.

Table 172: Possible Short Packets

Data Type	Description
0x00	Frame Start Code
0x01	Frame End Code
0x02	Line Start Code (Optional)
0x03	Line End Code (Optional)
0x04 – 0x07	Reserved

29.11.1.3 Long Packets

CSI long packets are data packets which always consist of 1 line of data per packet with exception of arbitrary data packets. There are various data types defined. Please refer to the Input Data Format section for a summary of CSI data formats and their corresponding CSI module internal output format.

Generic long packets need special handling. There are 3 generic long packet types defined: null packet (DT=0x10), blanking data packet (DT=0x11), and embedded data packet (DT=0x12). There are also 5 reserved generic long packet types (DT=0x13 to 0x17). The header parser should discard all null packets and all reserved generic long packets.

Blanking data packets may contain blank color or blank image information. The transmission of blanking data packet is optional in the CSI2.0 specification. Null and blanking data are defined in the CSI2.0 specification mainly to accommodate a receiver which is timing sensitive and requires line start/end for every line in the frame. As a CSI receiver, this product does not have such blank timing sensitivity and it typically does not process blank data and does not store it to memory. Blanking data packet can therefore be discarded by the header parser.

However, some modules may have a requirement for specific number of “blank” or extra lines at the end of vertical active area, such as the ISP module, which requires input active scan area to be about 6 lines larger than output active area. If a module that takes CSI output stream requires extra lines at the end of active image then blanking packet data can be used to generate these additional lines. In the future, there might be other reasons to process blank data. So, an option should be provided to accept and process blanking data. In the absence of better definition of blanking data format in the CSI specification, if software decides that blanking data cannot be discarded then, the pixel parser should assume that the blanking data format is the same as the programmed expected data type of the pixel stream.

The current CSI2.0 specification specifies that embedded data packets consists of 8-bit arbitrary data. This embedded data is, typically not the same pixel color as the image data. It might be used to send headers/status for JPEG/MPEG compressed data at the beginning or end of image data or in between image data lines. It might also be used to send closed caption text information or other type of information.

The exact use of embedded data is not clear, and therefore in most cases, embedded data is probably not used, or if sent by the transmitter, the embedded data packets can simply be discarded. If for some reason, it is important to receive the embedded data, there are other options that should be supported. Since there are two pixel parsers, if there is only one video source, it is best to direct embedded data to the other pixel parser which does not process the normal image data. In this way, the embedded data can be treated the same way as 8-bit arbitrary/compressed image data and can be output in 8-bit/clock or 16-bit/clock format. If there are two video sources that occupy both the pixel parsers, then embedded data if it has to be stored to memory, needs to be output in the same channel as the image data as 8-bit/16-bit data. But this also means that the module that receives output of CSI module and does image processing must have the ability to differentiate embedded data and not do image processing on this data prior to writing it to memory.

29.11.1.4 Top or Bottom Field Tag

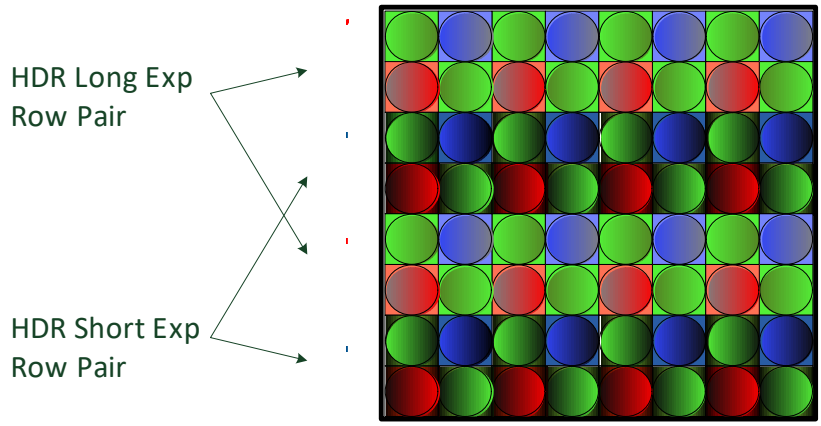
During frame start, a tag is passed from CSI to VI to indicate top or bottom field. The 1-bit tag is at the least significant bit of the CSI output bus. If the bit is “1”, the field is top, but bottom otherwise.

29.12 Test Pattern Generator (TPG)

The test pattern generator is a configurable resource introduced to improve hardware verification capability for the Tegra CSI. Because a provision for inserting pixel data from the Host interface is not available, the hardware verification of the CSI and VI units is difficult. The only sources of pixel data that can exercise the CSI are image sensors. In general, the sensor data cannot be controlled, making any self-checking test impossible. Test pattern generation modes offered by sensors are also ineffective because they have limited control.

The Tegra X1 CSI block introduces the AO-HDR feature in which data may be presented with alternate gain and exposure settings on each row as shown in the following figure.

Figure 120: Long and HDR Long and Short Exposure Rows Interleaved



The TPG will allow the scaling of the test patterns based on configuration settings. The first row pair is considered the long exposure row pair, and the second is the short exposure pair. In order to provide some flexibility, the option to either scale the short row pairs up or down is supported, which allows the simulation of HDR images while either allowing saturation or preventing it.

Table 173: TPG HDR Configuration Parameter

Register	Field	Bits	Format	Description
CSI_PG_AOHDR_[A..B]	ENABLE	0	U1	Controls whether the AT is operational. 0: DISABLED 1: ENABLED
	GAIN_RATIO	2:1	U2	Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ratio (short rows >> 2)
CSI1_PG_AOHDR_[A..B]	ENABLE	0	U1	Controls whether the AT is operational. 0: DISABLED 1: ENABLED
	GAIN_RATIO	2:1	U2	Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ratio (short rows >> 2)

Table 173: TPG HDR Configuration Parameter

Register	Field	Bits	Format	Description
CSI2_PG_AOHDR_[A..B]	ENABLE	0	U1	Controls whether the AT is operational. 0: DISABLED 1: ENABLED
	GAIN_RATIO	2:1	U2	Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ration (short rows >> 2)

29.12.1 TPG Limited Error Status

When enabled, the TPG will disable the following errors and associated interrupts for each instance of the CSI:

- Pixel Parser A:
  - HPA_UNC_HDR_ERR
  - PPA_PL_CRC_ERR
  - PPA_HDR_ERR_COR
- Pixel Parser B:
  - HPB_UNC_HDR_ERR
  - PPB_PL_CRC_ERR
  - PPB_HDR_ERR_COR

There are two separate test pattern generators for the CSI, CSI1, and CSI2 instances that can be configured to provide for the generation of synthetic image data, which is delivered to the PPA and PPB input FIFOs. The image data is multiplexed into the CSI, CSI1, and CSI2 datapaths between lane-merging logic and the data FIFOs. Programmable configuration parameters are available to allow variation in pixel data, data type, and spacing between various packets. In the real system, the LP state transition between any two CSI packets guarantees some idle time between the lines. This spacing between the lines from the pattern generator should be programmed to a minimum of 8 cycles, as the later stages of VI expects some idle cycles between the lines in order for it not to overflow.

A difference engine architecture is employed which can generate almost arbitrary patterns that consist of frequency sweeps or intensity ramps in both the horizontal and vertical directions. In addition, by using selected bits from the difference engine variables as addresses, a color patchwork pattern can be generated. The output from the difference engine is modeled off of a second-order partial differential equation of the form:

Where:

$$\varphi_{out} = \varphi + x \cdot \frac{d\varphi}{dx} + x^2 \cdot \frac{d^2\varphi}{dx^2}$$

$$\varphi = PHASE$$

$$\frac{d\varphi}{dx} = FREQ$$

$$\frac{d^2\varphi}{dx^2} = FREQ\_RATE$$

The PHASE, FREQ, and FREQ_RATE variables may be programmed to create a variety of different test patterns and are either used directly or as an index into a look-up table (LUT). The table below summarizes the configuration parameters for the TPG.

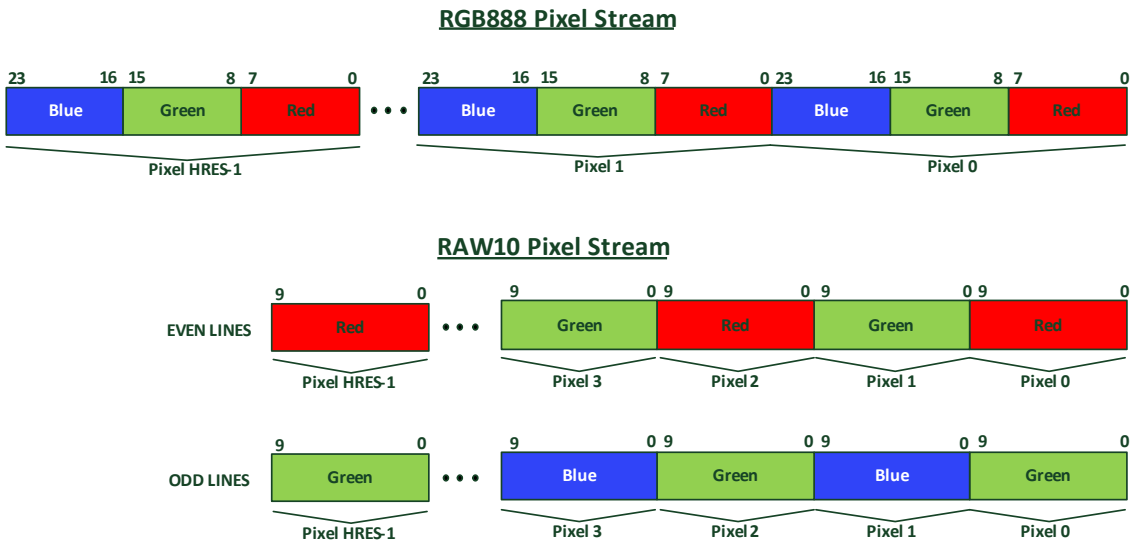
Table 174: Description of TPG Configuration Parameters

Configuration Variables for CSI_0, CSI1_0, CSI2_0 and CSI_1, CSI1_1, CSI2_1 Test Pattern Generators	TPG Frame Configuration	Image Height	Lines per frame
		Image Width	Pixels per line
		Vertical Blanking	Blank lines between frames
		Horizontal Blanking	Blank pixels between lines
		Initial Phase	Initial phase
		Mode	Direct or color patch
		Format	RGB888 or RAW10 support
	TPG Channel Configurations (R, G, and B)	Vertical Initial Frequency	Initial frequency
		Horizontal Initial Frequency	Initial frequency
		Vertical Frequency Rate	Rate of change for vertical frequency
		Horizontal Frequency Rate	Rate of change for horizontal frequency

29.12.2 TPG Frame Configuration Parameters

The TPG can be configured to provide the test images in either an RGB888 or Bayer RAW10 data format. The RGB888 data is delivered to the lane merging logic as a sequence of bytes representing the RGB triads for each pixel in an HRESxVRES image frame. For the Bayer RAW10 data, the color components of the RGB pixel are sampled in an alternating fashion, depending on whether the current row and pixel within the row is even or odd.

Figure 121: RGB888 and RAW10 TPG Pixel Streams



Note: *Byte ordering for RGB888 and RAW10 on the CSI2VI interface follows the same convention as data received over the MIPI interface.*

The data from the test pattern generator has the same format (with the packet header, CRC, ECC, etc.) as received on the output of lane-merging logic during normal operation.

29.12.3 Modes of Operation

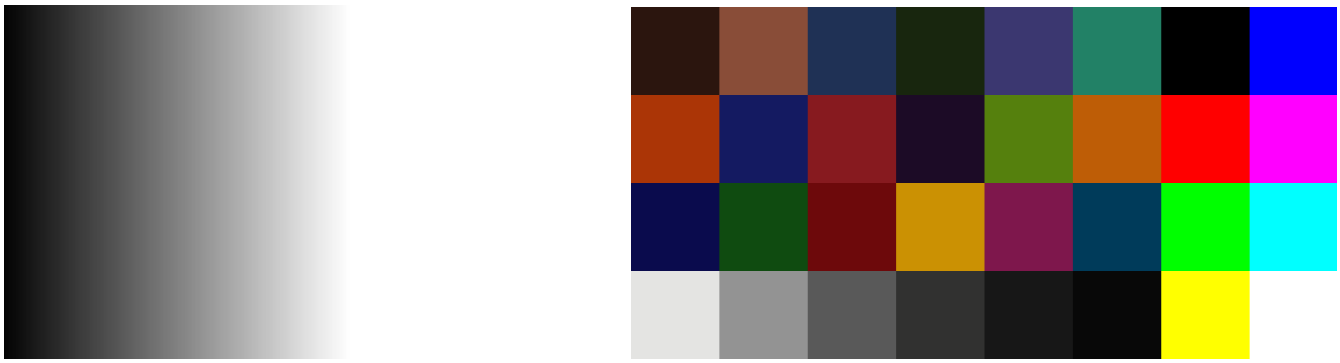
The test pattern generator operates in two modes:

- Direct — The output of the difference engine computation will be viewed directly.
- Color Patch — The difference engine output will be used to generate a patchwork of colors.

In Direct mode, the output comes directly from the value of the phase accumulator in the difference engine. An example of the kind of pattern that can be made is shown in (a) of the figure below. In Color Patch mode, the output comes from indexing a 32-element LUT.

The various bits from both the horizontal and vertical phase accumulators are used as address bits into a LUT of standard color values, allowing a patchwork of color test patterns to be generated. The colors in the pattern consist of linear RGB space representations of the 24 “Macbeth” test chart colors, shown on the left side of Figure 122(b), and the 8 “100% saturation” colors of a TV test pattern, on the right side of Figure 122(a). This gives a total of 32 color patches.

Figure 122: Example of (a) Direct Mode (b) Color Patch Generator Output



(a) (b)

29.13 Differential Pulse Code Modulation Support

The use of Differential Pulse Code Modulation (DPCM) compression schemes to reduce the data bandwidth requirements have been approached in both the SMIA and MIPI camera serial interface standards. The MIPI CSI-2 specification, v1.1, introduced a new data type for compressed data. In this format, RAW-10 and RAW-12 data can be transmitted as 6/7/8 bits per pixel. The SMIA also introduces a compression for RAW14 data providing transmission as 8/10 bits per pixel. The DPCM support is an optional feature, and the data compression is not mandated. However, if data compression is used, it must be implemented as described in annex E of the MIPI CSI-2 specification v01-01-00.

The MIPI and SMIA data compression schemes use an X–Y–Z naming convention, where:

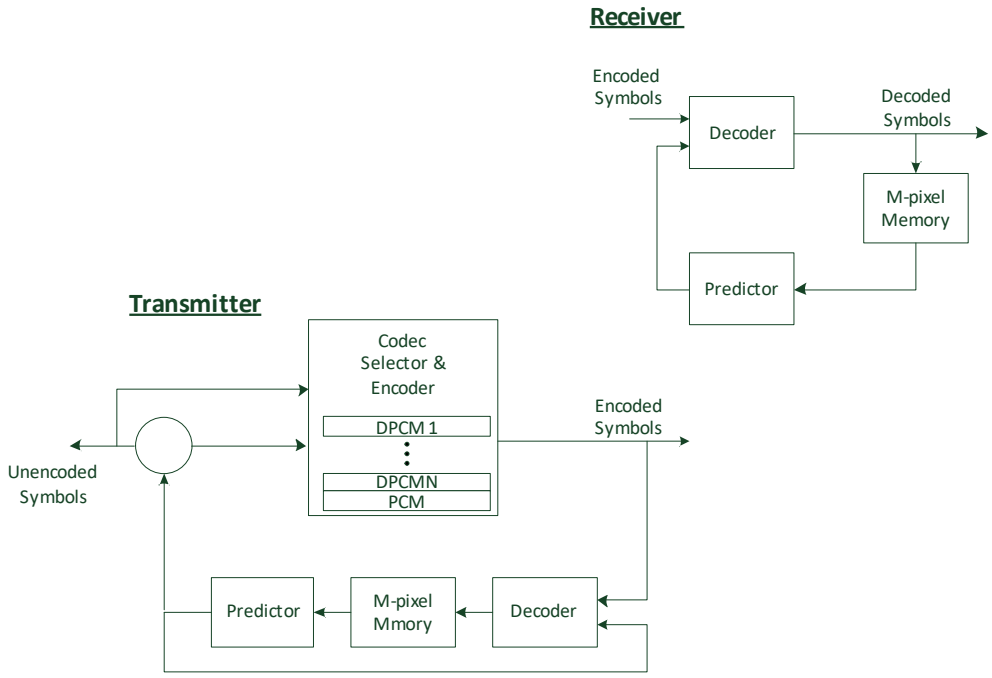
- X is the number of bits per pixel in the original image
- Y is the encoded (compressed) bits per pixel
- Z is the decoded (uncompressed) bits per pixel.

The following data compression schemes are defined for the Tegra X1 CSI unit:

- 14-bit RAW Compression Support (SMIA): 14-8-14; 14-10-14
- 12-bit RAW Compression Support (MIPI): 12–8–12; 12-7-12; 12-6-12
- 10-bit RAW Compression Support (MIPI): 10–8–10; 10-7-10; 10-6-10

To identify the type of data on the CSI-2 interface, packets with compressed data have a user-defined data type value. The encoder uses a simple algorithm to encode the pixel values. A fixed number of pixel values at the beginning of each line are encoded without using prediction. These first few values are used to initialize the predictor block. The remaining pixel values on the line are encoded using prediction. The following figure illustrates the data compression system for the transmitter and receiver.

Figure 123: Data Compression System Block Diagram



29.14 Software Requirements

29.14.1 Error Counters

To help debug, three 32-bit error counters have been implemented. Software can program each counter to increment at the event of any one of many error conditions or status change. The error conditions are:

- Header errors corrected
- Header uncorrectable errors
- CRC errors
- Packets too short, actual length less than WC
- Overflow errors due to padding packets too short
- FIFO overflow errors
- Illegal word count
- SoT single-bit error
- SoT multi-bit error, packet discarded
- EoT sequence error
- LP-CTRL error

The statistics that can be monitored include:

- Line packets processed
- Total packets processed

29.14.2 Programming Sequence

The following sequence is recommended for capturing a single frame:

1. Set up CSI registers for use case such as number of lanes, virtual channel, etc.
2. Initialize and power up CSI interface
3. Wait for initialization time or done signal from calibration logic
4. Power up camera through the I^2C interface
5. All CSI data and clock lanes are in stop state, LP11
6. Initiate frame capture through the I^2C
7. Frame done, CSI goes back to stop state, LP11

29.14.3 Escape Mode Handling

In the MIPI PHY specification, an escape mode is available for low speed command and data transfer. In our implementation, the escape mode entry is detected inside CSICIL. The command byte will be deposited in a register and interrupt generated from CSI. Software reads the register and decodes. CSI supports only one escape mode command which is Ultra-Low Power Mode (ULPM).

ULPM Command

Upon receiving the ULPM, software will synchronously power down the CSI interface and the camera as follows:

1. Software puts the camera in sleep mode through I^2C
2. The camera sends a ULPM command to CSI
3. CSI receives the ULPM command and raises an interrupt
4. Software powers down CSI, which goes to sleep in sync with the camera's CSI transmitter

29.15 DPHY Modes of Operation

The MIPI DPHY has 3 basic modes of operation. High speed mode employs differential signaling and achieves data rate of 1.5 Gbps. Both the driver and receiver are matched to 100 ohms differential. The driver is voltage mode for lower power consumption as opposed to current mode.

Low power control mode employs single-ended CMOS signaling for handshaking between camera and host. In this mode, there is no clock and no maximum symbol time defined in the specification. The receiver samples the input using both edges of the internal pixel clock, so the timing resolution is half the clock period of the pixel clock period.

Table 175: MIPI DPHY Mode of Operations

Modes	Description	Clock
High speed (HS)	High speed differential signaling. Up to 1.5 Gbps. Burst transmission for low power.	750 MHz differential
Low Power (LP) Control	Single-ended 1.2V CMOS level. Low speed signaling for handshaking. Supports ULPM sleep state.	No Clock

29.16 MIPI-CSI Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

29.16.1 CSI_INPUT_STREAM_A_CONTROL_0

CSI Input Stream A Control

Offset: 0x20e | Byte Offset: 0x838 | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_A_SKIP_PACKET_THRESHOLD: CSI-A Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_A_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE
2	0x1	CSI_A_BYPASS_ALIGN: Bypass aligning CSIA and CSIB lanes if the valids for the lanes are out of sync by one cycle
1:0	0x0	CSI_A_DATA_LANE: CSI-A Data Lane. 0= 1 data lane; 1= 2 data lanes; 2= 3 data lanes; 3= 4 data lanes Note: Three data lanes are not supported in Test Pattern Generation mode.

29.16.2 CSI_PIXEL_STREAM_A_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x20f | Byte Offset: 0x83c | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

Bit	Reset	Description
29:28	X	CSI_PPA_PAD_FRAME: CSI Pixel Parser A Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPA_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPA_WORD_COUNT needs to be set to the number of input bytes in each line's payload. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.
27	X	CSI_PPA_HEADER_EC_ENABLE: CSI Pixel Parser A Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected, or not. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser A. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB).
25:24	X	CSI_PPA_PAD_SHORT_LINE: CSI Pixel Parser A Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count). 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: A short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receives CSI output stream to hang up.

Bit	Reset	Description
21:20	X	CSI_PPA_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser A Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPA_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by other CSI pixel stream processor. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED: Output embedded data as an 8-bpp arbitrary data stream.
19:16	X	CSI_PPA_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser A Output Format Options. This parameter specifies options for output data format. 0 = ARBITRARY: Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE: Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444 will be zeroed.
8	0x1	CSI_PPA_WC_CHECK: CSI Pixel Parser A Data WC Check. This parameter specifies whether the word count is checked. 0 = DISABLE: Wordcount Check is disabled 1 = ENABLE: Wordcount Check is enabled.
7	X	CSI_PPA_CRC_CHECK: CSI Pixel Parser A Data CRC Check. This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.
6	X	CSI_PPA_WORD_COUNT_SELECT: CSI Pixel Parser A Word Count Select. This parameter is effective only if packet header is sent as part of the stream. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/ packet is specified by CSI_PPA_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPA_WORD_COUNT is different than the count in the packet header. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. It is recommended to always program this bit to HEADER.
5	X	CSI_PPA_DATA_IDENTIFIER: CSI Pixel Parser A Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPA_DATA_TYPE and against CSI_PPA_VIRTUAL_CHANNEL_ID). In this case, CSI_PPA_DATA_TYPE specifies the stream data format. 1 = ENABLED: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPA_DATA_TYPE and the CSI_PPA_VIRTUAL_CHANNEL_ID.
4	X	CSI_PPA_PACKET_HEADER: CSI Pixel Parser A Packet Header processing. This specifies whether packet header is sent in the beginning of packet or not. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPA_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.
2:0	0x0	CSI_PPA_STREAM_SOURCE: CSI Pixel Parser A Stream Source 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B

29.16.3 CSI_PIXEL_STREAM_A_CONTROL1_0

CSI Pixel Stream A Control 1

Offset: 0x210 | Byte Offset: 0x840 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)|

Bit	Reset	Description
7:4	0x0	CSI_PPA_TOP_FIELD_FRAME_MASK: CSI Pixel Parser A Top Field Frame Mask

Bit	Reset	Description
3:0	0x0	CSI_PPA_TOP_FIELD_FRAME: CSI Pixel Parser A Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise OR of ~(CSI_PPA_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPA_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet. Due to the reliance on WC field in packet header, interlaced support will not work if WC is selected from REGISTER mode.

29.16.4 CSI_PIXEL_STREAM_A_GAP_0

CSI Pixel Stream A Gap

Offset: 0x211 | Byte Offset: 0x844 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPA_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that the minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPA_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that the minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

29.16.5 CSI_PIXEL_STREAM_PPA_COMMAND_0

CSI Pixel Parser A Command

Offset: 0x212 | Byte Offset: 0x848 | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPA_START_MARKER_FRAME_MAX: CSI Pixel Parser A Start Marker Maximum. Start Frame is indicated when the Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.
11:8	X	CSI_PPA_START_MARKER_FRAME_MIN: CSI Pixel Parser A Start Marker Minimum. Start Frame is indicated when the Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.
4	X	CSI_PPA_VSYNC_START_MARKER: CSI Pixel Parser A VSYNC Start Marker. The start of frame is indicated when the VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC. 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPA_START_MARKER_FRAME_MIN and less than or equal to CSI_PPA_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC
2	0x1	CSI_PPA_SINGLE_SHOT: CSI Pixel Parser A Single Shot Mode. Software should clear it along with disabling the CSI_PPA_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPA_ENABLE: CSI Pixel Parser A Enable. This parameter controls CSI Pixel Parser A to start or stop receiving data. Reset (disable immediately) Enabling the pixel parser does not enable the corresponding input source to receive data. If the pixel parser is enabled later than the corresponding input source, the CSI will keep on rejecting incoming stream, until it encounters a valid SF. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST

29.16.6 CSI_PIXEL_STREAM_A_EXPECTED_FRAME_0

CSI Pixel Stream A Expected Frame

Offset: 0x213 | Byte Offset: 0x84c | Read/Write: R/W | Reset: 0x0000XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:4	X	PPA_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	0x0	PPA_ENABLE_LINE_TIMEOUT: When set to one enables checking of the time between start line requests from the Header Parser to CSI-PPA. A fake EF will be outputted by CSI-PPA if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines outputted, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPA_PAD_FRAME.

29.16.7 CSI_CSI_PIXEL_PARSER_A_INTERRUPT_MASK_0

CSI Pixel Parser A Interrupt Mask

Offset: 0x214 | Byte Offset: 0x850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPA_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPA_UNC_HDR_ERR. 0 = DISABLED: Do not generate an interrupt when HPA_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPA_UNC_HDR_ERR is set.
10	0x0	PPA_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPA_SPARE_STATUS_1. 0 = DISABLED: Do not generate an interrupt when PPA_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPA_SPARE_STATUS_1 is set.
9	0x0	PPA_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPA_INTERFRAME_LINE. 0 = DISABLED: Do not generate an interrupt when PPA_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPA_INTERFRAME_LINE is set.
8	0x0	PPA_EXTRA_SF_INT_MASK: Interrupt Mask for PPA_EXTRA_SF. 0 = DISABLED: Do not generate an interrupt when PPA_EXTRA_SF is set. 1 = ENABLED: Generate an interrupt when PPA_EXTRA_SF is set.
7	0x0	PPA_SHORT_FRAME_INT_MASK: Interrupt Mask for PPA_SHORT_FRAME. 0 = DISABLED: Do not generate an interrupt when PPA_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPA_SHORT_FRAME is set.
6	0x0	PPA_STMERR_INT_MASK: Interrupt Mask for PPA_STMERR. 0 = DISABLED: Do not generate an interrupt when PPA_STMERR is set. 1 = ENABLED: Generate an interrupt when PPA_STMERR is set.
5	0x0	PPA_FIFO_OVRF_INT_MASK: Interrupt Mask for PPA_FIFO_OVRF. 0 = DISABLED: Do not generate an interrupt when PPA_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPA_FIFO_OVRF is set.
4	0x0	PPA_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPA_PL_CRC_ERR. 0 = DISABLED: Do not generate an interrupt when PPA_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPA_PL_CRC_ERR is set.
3	0x0	PPA_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPA_SL_PKT_DROPPED. 0 = DISABLED: Do not generate an interrupt when PPA_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PKT_DROPPED is set.
2	0x0	PPA_SL_PROCESSED_INT_MASK: Interrupt Mask for PPA_SL_PROCESSED. 0 = DISABLED: Do not generate an interrupt when PPA_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PROCESSED is set.
1	0x0	PPA_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPA_ILL_WD_CNT. 0 = DISABLED: Do not generate an interrupt when PPA_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPA_ILL_WD_CNT is set.
0	0x0	PPA_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPA_HDR_ERR_COR. 0 = DISABLED: Do not generate an interrupt when PPA_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPA_HDR_ERR_COR is set.

29.16.8 CSI_CSI_PIXEL_PARSER_A_STATUS_0

Pixel Parser A Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_PIXEL_PARSER_STATUS will clear only PPA_ILL_WD_CNT.

Offset: 0x215 | Byte Offset: 0x854 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	HPA_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser A parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected. The packet will be discarded.
10	X	PPA_SPARE_STATUS_1: PPA Spare Status bit. This bit will get set when Pixel Parser A has a line timeout. Line timeout needs to be enabled by setting PPA_ENABLE_LINE_TIMEOUT and programming PPA_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPA_INTERFRAME_LINE: Set when CSI-PPA receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.
8	X	PPA_EXTRA_SF: Set when CSI-PPA receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPA will insert a fake EF and the drop the current frame with Correct SF.
7	X	PPA_SHORT_FRAME: Set when CSI-PPA receives a short frame. This bit gets set even if CSI_PPA_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPA_STMERR: Stream Error, set when the control output of PPA does not follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
5	X	PPA_FIFO_OVRF: FIFO Overflow, set when the FIFO that is feeding packets to PPA overflows.
4	X	PPA_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPA had a payload CRC error.
3	X	PPA_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_A_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPA_SL_PROCESSED: Short Line Processed, Set when a line with a payload that is shorter than its packet header word count is processed by PPA.
1	X	PPA_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that does not generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPA.
0	X	PPA_HDR_ERR_COR: Header Error Corrected, Set when a packet that was processed by PPA has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_A_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

29.16.9 CSI_CSI_SW_SENSOR_A_RESET_0

Offset: 0x216 | Byte Offset: 0x858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_A_RESET: Reset CSI sensor A

29.16.10 CSI_INPUT_STREAM_B_CONTROL_0

CSI Input Stream B Control

Offset: 0x21b | Byte Offset: 0x86c | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_B_SKIP_PACKET_THRESHOLD: CSI-B Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).

Bit	Reset	Description
4	0x0	CSI_B_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE: Skip packet feature is enabled.
2	0x1	CSI_B_BYPASS_ALIGN: Bypass aligning CSIB lanes if valids for the lanes are out of sync by a cycle
1:0	0x0	CSI_B_DATA_LANE: CSI-B Data Lane 0= 1 data lane, 1= 2 data lanes, 2= 3 data lanes (not supported), 3= 4 data lanes (not supported) Note: 3 data lanes is not supported in Test Pattern Generation mode.

29.16.11 CSI_PIXEL_STREAM_B_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x21c | Byte Offset: 0x870 | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

Bit	Reset	Description
29:28	X	CSI_PPB_PAD_FRAME: CSI Pixel Parser B Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPB_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPB_WORD_COUNT needs to be set to the number of input bytes in each lines payload. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.
27	X	CSI_PPB_HEADER_EC_ENABLE: CSI Pixel Parser B Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected or not. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser B. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB).
25:24	X	CSI_PPB_PAD_SHORT_LINE: CSI Pixel Parser B Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count) short line is not padded (will output less pixels than expected). 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: This option is not recommended and may cause other modules that receive CSI output stream to hang up.
21:20	X	CSI_PPB_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser B Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPB_DATA_TYPE is not embedded data and that embedded data is not already processed by another CSI pixel stream processor. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED: Output embedded data as 8-bpp arbitrary data stream.
19:16	X	CSI_PPB_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser B Output Format Options. This parameter specifies output data format. 0 = ARBITRARY: Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS. color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE: Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed.

Bit	Reset	Description
8	0x1	CSI_PPB_WC_CHECK: CSI Pixel Parser B Data WC Check. This parameter specifies whether the wordcount is checked. 0 = DISABLE: Wordcount Check is disabled 1 = ENABLE: Wordcount Check is enabled
7	X	CSI_PPB_CRC_CHECK: CSI Pixel Parser B Data CRC Check. This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.
6	X	CSI_PPB_WORD_COUNT_SELECT: CSI Pixel Parser B Word Count Select. This parameter is effective only if packet header is sent as part of the stream. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/ packet is specified by CSI_PPB_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPB_WORD_COUNT is different than the count in the packet header. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. It is recommended to always program this bit to HEADER.
5	X	CSI_PPB_DATA_IDENTIFIER: CSI Pixel Parser B Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPB_DATA_TYPE and against CSI_PPB_VIRTUAL_CHANNEL_ID). In this case, CSI_PPB_DATA_TYPE specifies the stream data format. 1 = ENABLED: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPB_DATA_TYPE and the CSI_PPB_VIRTUAL_CHANNEL_ID.
4	X	CSI_PPB_PACKET_HEADER: CSI Pixel Parser B Packet Header processing. This specifies whether packet header is sent in the beginning of packet or not. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPB_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.
2:0	0x0	CSI_PPB_STREAM_SOURCE: CSI Pixel Parser B Stream Source 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B

29.16.12 CSI_PIXEL_STREAM_B_CONTROL1_0

CSI Pixel Stream B Control 1

Offset: 0x21d | Byte Offset: 0x874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	CSI_PPB_TOP_FIELD_FRAME_MASK: CSI Pixel Parser B Top Field Frame Mask
3:0	0x0	CSI_PPB_TOP_FIELD_FRAME: CSI Pixel Parser B Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise OR of ~(CSI_PPB_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPB_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet. Due to the reliance on WC field in packet header, interlaced support will not work if WC is selected from REGISTER mode.

29.16.13 CSI_PIXEL_STREAM_B_GAP_0

CSI Pixel Stream B Gap

Offset: 0x21e | Byte Offset: 0x878 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPB_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that the minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPB_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that the minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

29.16.14 CSI_PIXEL_STREAM_PPB_COMMAND_0

CSI Pixel Parser B Command

Offset: 0x21f | Byte Offset: 0x87c | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPB_START_MARKER_FRAME_MAX: CSI Pixel Parser B Start Marker Maximum. Start Frame is indicated when the Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.
11:8	X	CSI_PPB_START_MARKER_FRAME_MIN: CSI Pixel Parser B Start Marker Minimum. Start Frame is indicated when the Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.
4	X	CSI_PPB_VSYNC_START_MARKER: CSI Pixel Parser B VSYNC Start Marker. 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPB_START_MARKER_FRAME_MIN and less than or equal to CSI_PPB_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC: Start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC.
2	0x1	CSI_PPB_SINGLE_SHOT: CSI Pixel Parser B Single Shot Mode. Software should clear it along with disabling CSI_PPB_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPB_ENABLE: CSI Pixel Parser B Enable. This parameter controls CSI Pixel Parser B to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST: reset (disable immediately). Enabling the pixel parser does not enable the corresponding input source to receive data. If the pixel parser is enabled later than the corresponding input source, the CSI will keep rejecting the incoming stream, until it encounters a valid SF.

29.16.15 CSI_PIXEL_STREAM_B_EXPECTED_FRAME_0

CSI Pixel Stream B Expected Frame

Offset: 0x220 | Byte Offset: 0x880 | Read/Write: R/W | Reset: 0x0000XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:4	X	PPB_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.

Bit	Reset	Description
0	0x0	PPB_ENABLE_LINE_TIMEOUT: When set to one, enables checking of the time between start line requests from the Header Parser to CSI-PPB. A fake EF will be output by CSI-PPB if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines output, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPB_PAD_FRAME.

29.16.16 CSI_CSI_PIXEL_PARSER_B_INTERRUPT_MASK_0

CSI Pixel Parser B Interrupt Mask

Offset: 0x221 | Byte Offset: 0x884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPB_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPB_UNC_HDR_ERR. 0 = DISABLED: Do not generate an interrupt when HPB_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPB_UNC_HDR_ERR is set.
10	0x0	PPB_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPB_SPARE_STATUS_1. 0 = DISABLED: Do not generate an interrupt when PPB_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPB_SPARE_STATUS_1 is set.
9	0x0	PPB_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPB_INTERFRAME_LINE. 0 = DISABLED: Do not generate an interrupt when PPB_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPB_INTERFRAME_LINE is set.
8	0x0	PPB_EXTRA_SF_INT_MASK: Interrupt Mask for PPB_EXTRA_SF. 0 = DISABLED: Do not generate an interrupt when PPB_EXTRA_SF is set. 1 = ENABLED: Generate an interrupt when PPB_EXTRA_SF is set.
7	0x0	PPB_SHORT_FRAME_INT_MASK: Interrupt Mask for PPB_SHORT_FRAME. 0 = DISABLED: Do not generate an interrupt when PPB_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPB_SHORT_FRAME is set.
6	0x0	PPB_STMERR_INT_MASK: Interrupt Mask for PPB_STMERR. 0 = DISABLED: Do not generate an interrupt when PPB_STMERR is set. 1 = ENABLED: Generate an interrupt when PPB_STMERR is set.
5	0x0	PPB_FIFO_OVRF_INT_MASK: Interrupt Mask for PPB_FIFO_OVRF. 0 = DISABLED: Do not generate an interrupt when PPB_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPB_FIFO_OVRF is set.
4	0x0	PPB_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPB_PL_CRC_ERR. 0 = DISABLED: Do not generate an interrupt when PPB_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPB_PL_CRC_ERR is set.
3	0x0	PPB_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPB_SL_PKT_DROPPED. 0 = DISABLED: Do not generate an interrupt when PPB_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PKT_DROPPED is set.
2	0x0	PPB_SL_PROCESSED_INT_MASK: Interrupt Mask for PPB_SL_PROCESSED. 0 = DISABLED: Do not generate an interrupt when PPB_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PROCESSED is set.
1	0x0	PPB_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPB_ILL_WD_CNT. 0 = DISABLED: Do not generate an interrupt when PPB_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPB_ILL_WD_CNT is set.
0	0x0	PPB_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPB_HDR_ERR_COR. 0 = DISABLED: Do not generate an interrupt when PPB_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPB_HDR_ERR_COR is set.

29.16.17 CSI_CSI_PIXEL_PARSER_B_STATUS_0

Pixel Parser B Status

Offset: 0x222 | Byte Offset: 0x888 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	HPB_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser B parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected.

Bit	Reset	Description
10	X	PPB_SPARE_STATUS_1: PPB Spare Status bit. This bit will get set when Pixel Parser B has a line timeout. Line timeout needs to be enabled by setting PPB_ENABLE_LINE_TIMEOUT and programming PPB_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPB_INTERFRAME_LINE: Set when CSI-PPB receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.
8	X	PPB_EXTRA_SF: Set when CSI-PPB receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPB will insert a fake EF and the drop the current frame with Correct SF.
7	X	PPB_SHORT_FRAME: Set when CSI-PPB receives a short frame. This bit gets set even if CSI_PPB_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPB_STMERR: Stream Error, set when the control output of PPB doesn't follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
5	X	PPB_FIFO_OVRF: FIFO Overflow, set when the FIFO that is feeding packets to PPB overflows.
4	X	PPB_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPB had a payload CRC error.
3	X	PPB_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_B_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPB_SL_PROCESSED: Short Line Processed, Set when a line with a payload that is shorter than its packet header word count is processed by PPB.
1	X	PPB_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that doesn't generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPB.
0	X	PPB_HDR_ERR_COR: Header Error Corrected, set when a packet that was processed by PPB has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_B_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

29.16.18 CSI_CSI_SW_SENSOR_B_RESET_0

Offset: 0x223 | Byte Offset: 0x88c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_B_RESET: Reset CSI sensor B

29.16.19 CSI_PHY_CIL_COMMAND_0

CSI PHY and CIL Command

Offset: 0x242 | Byte Offset: 0x908 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx00xxxxxx00)

Bit	Reset	Description
9:8	0x0	CSI_B_PHY_CIL_ENABLE: CSI B PHY and CIL Enable. This parameter controls CSI B PHY and CIL receiver to start or stop receiving data. disable (reset) 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE
1:0	0x0	CSI_A_PHY_CIL_ENABLE: CSI A PHY and CIL Enable. This parameter controls CSI A PHY and CIL receiver to start or stop receiving data. disable (reset) 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE

29.16.20 CSI_CIL_PAD_CONFIG0_0

CIL Pad Configuration 0

Offset: 0x243 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000xxxxxxxx)

Bit	Reset	Description
15:8	0x0	PAD_CIL_SPARE: Spare bit for CIL BIAS Config. PAD_CIL_SPARE[7] is used is being used to flush VI's Y-FIFO when it is being use as a stream source for one of the Pixel Parsers. Setting PAD_CIL_SPARE[7] to 1 will hold vi2csi_host_stall low. This will force VI's Y-FIFO to be purged. PAD_CIL_SPARE[7] must be low for the pixel parser to receive source data from VI's Y-FIFO.

29.16.21 CSI_CILA_PAD_CONFIG0_0

CIL-A Pad Configuration 0

Offset: 0x24b | Byte Offset: 0x92c | Read/Write: R/W | Reset: 0x00010007 (0b0000x000000000010000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILA_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. will cause stress and need waiver to use
30:28	0x0	PAD_CILA_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
26:24	0x0	PAD_CILA_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
23:21	0x0	PAD_CILA_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm This control is unused for CSI MIPI pads, which are receive only
20:18	0x0	PAD_CILA_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm This control is unused for CSI MIPI pads, which are receive only
17:16	0x1	PAD_AB_BK_MODE: 00: two independent 2x bricks 01: one 4x brick, received clock from partition A is used. Clock from partition B is not used 10: one 4x brick, received clock from partition B is used. Clock from partition A is not used 11: illegal
15	0x0	PAD_CILA_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILA_INADJ1: bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILA_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILA_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILA_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
2	0x1	PAD_CILA_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
1:0	0x3	PAD_CILA_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

29.16.22 CSI_CILA_PAD_CONFIG1_0

CIL-A Pad Configuration 4

Offset: 0x24c | Byte Offset: 0x930 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILA_SPARE_RO: Spare Read only bits for CILA Config
15:8	RW	0x0	PAD_CILA_SPARE: Spare bits for CILA Config. PAD_CILA_SPARE[15] is being used to disable the CSI-A RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled 1: push blocking enabled
7:6	RW	0x0	PAD_CILA_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only
5:4	RW	0x0	PAD_CILA_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only
3:2	RW	0x0	PAD_CILA_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only
1:0	RW	0x0	PAD_CILA_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only

29.16.23 CSI_PHY_CILA_CONTROL0_0

CSI-A PHY and CIL Control

Offset: 0x24d | Byte Offset: 0x934 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILA_CLK_SETTLE: Settle time for clock lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (72 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 csicil cycles for the default value. So the programming value (0 to 15) of this field works like this: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILA_BYPASS_LP_SEQ: The LP signals should sequence through LP11->LP01->LP00 state, to indicate to CLOCK CIL about the mode switching to HS Rx mode. In case the Camera is enabled earlier than CIL, it is highly likely that the camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
5:0	0x2	CILA_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field. 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

29.16.24 CSI_CSI_CIL_A_INTERRUPT_MASK_0

CSI Control and Interface Logic A Interrupt Mask

Offset: 0x24e | Byte Offset: 0x938 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
9	0x0	CILA_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CLK_CTRL_ERR. 0 = DISABLED: Do not generate an interrupt when CILA_CLK_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CLK_CTRL_ERR is set.
6	0x0	CILA_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILA_ESC_DATA_REC. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_DATA_REC is set.
5	0x0	CILA_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILA_ESC_CMD_REC. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_CMD_REC is set.
4	0x0	CILA_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CTRL_ERR. 0 = DISABLED: Do not generate an interrupt when CILA_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CTRL_ERR is set.
2	0x0	CILA_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILA_SYNC_ESC_ERR. 0 = DISABLED: Do not generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SYNC_ESC_ERR is set.
1	0x0	CILA_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_MB_ERR. 0 = DISABLED: Do not generate an interrupt when CILA_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_MB_ERR is set.
0	0x0	CILA_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_SB_ERR. 0 = DISABLED: Do not generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_SB_ERR is set.

29.16.25 CSI_CSI_CIL_A_STATUS_0

CSI Control and Interface Logic A Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_A_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 0x24f | Byte Offset: 0x93c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILA_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-A receives an Escape Mode Data byte. The Data Byte can be read from bits 7-0 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILA_ESC_CMD_REC is set.
5	X	CILA_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-A receives an Escape Mode Command byte. The Command Byte can be read from bits 7-0 of ESCAPE_MODE_COMMAND.
4	X	CILA_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILA_SYNC_ESC_ERR: Sync Escape Error, set when CIL-A detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
1	X	CILA_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	X	CILA_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

29.16.26 CSI_CSI_CILA_STATUS_0

CSI-CILA Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x250 | Byte Offset: 0x940 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILA_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILA_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILA_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-A for processing.
6	X	CILA_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILA_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILA_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-A for processing.
0	X	CILA_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-A detects incorrect line state sequence on clk lane

29.16.27 CSI_CIL_A_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-A.

Offset: 0x251 | Byte Offset: 0x944 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILA_ESC_CMD_BYTE: CIL-A Escape Mode Command Byte. This is the 8 bit entry command that was received, by CIL-A, during the last escape Mode sequence. CIL-A monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILA_ESC_CMD_REC status bit is set.

29.16.28 CSI_CIL_A_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-A.

Offset: 0x252 | Byte Offset: 0x948 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILA_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte. When read this field returns the last Escape Mode Data byte that was received by CIL-A. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILA_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte comes in.

29.16.29 CSI_CSICIL_SW_SENSOR_A_RESET_0

Offset: 0x253 | Byte Offset: 0x94c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_A_RESET: Reset CSICIL sensor A

29.16.30 CSI_CILB_PAD_CONFIG0_0

CIL-B Pad Configuration 0

Offset: 0x258 | Byte Offset: 0x960 | Read/Write: R/W | Reset: 0x00000007 (0b0000x000000000xx0000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILB_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. will cause stress and need waiver to use
30:28	0x0	PAD_CILB_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
26:24	0x0	PAD_CILB_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
23:21	0x0	PAD_CILB_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm This control is unused for CSI MIPI pads, which are receive only
20:18	0x0	PAD_CILB_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm This control is unused for CSI MIPI pads, which are receive only
15	0x0	PAD_CILB_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILB_INADJ1: bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILB_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILB_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILB_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
2	0x1	PAD_CILB_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
1:0	0x3	PAD_CILB_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

29.16.31 CSI_CILB_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 0x259 | Byte Offset: 0x964 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILB_SPARE_RO: Spare Read only bits for CILB Config
15:8	RW	0x0	PAD_CILB_SPARE: Spare bits for CILB Config. PAD_CILB_SPARE[15] is being used to disable the CSI-B RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled, 1: push blocking enabled
7:6	RW	0x0	PAD_CILB_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only
5:4	RW	0x0	PAD_CILB_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only
3:2	RW	0x0	PAD_CILB_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only
1:0	RW	0x0	PAD_CILB_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max This control is unused for CSI MIPI pads, which are receive only

29.16.32 CSI_PHY_CILB_CONTROL0_0

CSI-B PHY and CIL Control

Offset: 0x25a | Byte Offset: 0x968 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILB_CLK_SETTLE: Settle time for clock lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (72 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 CSICIL cycles for default value. So the programming value (0 to 15) of this field works like this: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILB_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
5:0	0x2	CILB_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field. 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

29.16.33 CSI_CSI_CIL_B_INTERRUPT_MASK_0

CSI Control and Interface Logic B Interrupt Mask

Offset: 0x25b | Byte Offset: 0x96c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
6	0x0	CILB_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILB_ESC_DATA_REC. 0 = DISABLED: Do not generate an interrupt when CILB_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_DATA_REC is set.
5	0x0	CILB_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILB_ESC_CMD_REC. 0 = DISABLED: Do not generate an interrupt when CILB_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_CMD_REC is set.
4	0x0	CILB_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CTRL_ERR. 0 = DISABLED: Do not generate an interrupt when CILB_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_CTRL_ERR is set.
2	0x0	CILB_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILB_SYNC_ESC_ERR. 0 = DISABLED: Do not generate an interrupt when CILB_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SYNC_ESC_ERR is set.
1	0x0	CILB_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_MB_ERR. 0 = DISABLED: Do not generate an interrupt when CILB_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_MB_ERR is set.
0	0x0	CILB_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_SB_ERR. 0 = DISABLED: Do not generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_SB_ERR is set.

29.16.34 CSI_CSI_CIL_B_STATUS_0

CSI Control and Interface Logic B Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_B_STATUS will clear only CILB_SOT_MB_ERR.

Offset: 0x25c | Byte Offset: 0x970 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILB_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-B receives an Escape Mode Data byte. The Data Byte can be read from bits 23-16 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILB_ESC_CMD_REC is set.
5	X	CILB_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-B receives an Escape Mode Command byte. The Command Byte can be read from bits 23-16 of ESCAPE_MODE_COMMAND.
4	X	CILB_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILB_SYNC_ESC_ERR: Sync Escape Error, set when CIL-B detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
1	X	CILB_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	X	CILB_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-B detects a single bit error in one of the packets start of transmission bytes. The packet will be sent to CSI-B for processing.

29.16.35 CSI_CSI_CILB_STATUS_0

CSI-CILB Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x25d | Byte Offset: 0x974 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILB_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILB_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILB_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-B for processing.
6	X	CILB_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILB_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILB_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-B detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-B for processing.
0	X	CILB_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-B detects incorrect line state sequence on clk lane

29.16.36 CSI_CIL_B_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-B.

Offset: 0x25e | Byte Offset: 0x978 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILB_ESC_CMD_BYTE: CIL-B Escape Mode Command Byte. This is the 8-bit entry command that was received, by CIL-B, during the last escape Mode sequence. CIL-B monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILB_ESC_CMD_REC status bit is set.

29.16.37 CSI_CIL_B_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-B.

Offset: 0x25f | Byte Offset: 0x97c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILB_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte. When read this field returns the last Escape Mode Data byte that was received by CIL-B. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILB_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte comes in.

29.16.38 CSI_CSICIL_SW_SENSOR_B_RESET_0

Offset: 0x260 | Byte Offset: 0x980 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_B_RESET: Reset CSICIL sensor B

29.16.39 CSI_PATTERN_GENERATOR_CTRL_A_0

Offset: 0x271 | Byte Offset: 0x9c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_A: Mode for Sensor A 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_A: Automatic phase increment mode for sensor A
0	0x0	PG_ENABLE_A: Enable Pattern Generator for sensor A

29.16.40 CSI_PG_BLANK_A_0

Offset: 0x272 | Byte Offset: 0x9c8 | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_A: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_A: Horizontal Blanking for PG

29.16.41 CSI_PG_PHASE_A_0

Offset: 0x273 | Byte Offset: 0x9cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_A: Initial Phase

29.16.42 CSI_PG_RED_FREQ_A_0

Offset: 0x274 | Byte Offset: 0x9d0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.43 CSI_PG_RED_FREQ_RATE_A_0

Offset: 0x275 | Byte Offset: 0x9d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.44 CSI_PG_GREEN_FREQ_A_0

Offset: 0x276 | Byte Offset: 0x0x9d8 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.45 CSI_PG_GREEN_FREQ_RATE_A_0

Offset: 0x277 | Byte Offset: 0x9dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.46 CSI_PG_BLUE_FREQ_A_0

Offset: 0x278 | Byte Offset: 0x9e0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.47 CSI_PG_BLUE_FREQ_RATE_A_0

Offset: 0x279 | Byte Offset: 0x9e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.48 CSI_PG_AOHDR_A_0

Offset: 0x27a | Byte Offset: 0x9e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:1	0x0	PG_AOHDR_GAIN_RATIO_A: Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ration (short rows >> 2)
0	0x0	PG_AOHDR_ENABLE_A: AOHDR enable

29.16.49 CSI_PATTERN_GENERATOR_CTRL_B_0

Offset: 0x27e | Byte Offset: 0x9f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_B: Mode for Sensor B 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_B: Automatic phase increment mode for sensor B
0	0x0	PG_ENABLE_B: Enable Pattern Generator for sensor B

29.16.50 CSI_PG_BLANK_B_0

Offset: 0x27f | Byte Offset: 0x9fc | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_B: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_B: Horizontal Blanking for PG

29.16.51 CSI_PG_PHASE_B_0

Offset: 0x280 | Byte Offset: 0xa00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_B: Initial Phase

29.16.52 CSI_PG_RED_FREQ_B_0

Offset: 0x281 | Byte Offset: 0xa04 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.53 CSI_PG_RED_FREQ_RATE_B_0

Offset: 0x282 | Byte Offset: 0xa08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.54 CSI_PG_GREEN_FREQ_B_0

Offset: 0x283 | Byte Offset: 0xa0c | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.55 CSI_PG_GREEN_FREQ_RATE_B_0

Offset: 0x284 | Byte Offset: 0xa10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.56 CSI_PG_BLUE_FREQ_B_0

Offset: 0x285 | Byte Offset: 0xa14 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.57 CSI_PG_BLUE_FREQ_RATE_B_0

Offset: 0x286 | Byte Offset: 0xa18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.58 CSI_PG_AOHDR_B_0

Offset: 0x287 | Byte Offset: 0xa1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:1	0x0	PG_AOHDR_GAIN_RATIO_B: Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ration (short rows >> 2)
0	0x0	PG_AOHDR_ENABLE_B: AOHDR enable

29.16.59 CSI_DPCM_CTRL_A_0

Offset: 0x28b | Byte Offset: 0xa2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_A: DPCM A compression ratio 0: BYPASS 1: 10_8_10 2: 10_7_10 3: 10_6_10 4: 12_8_12 5: 12_7_12 6: 12_6_12 7: 14_10_14 8: 14_8_14
0	0x0	DPCM_PREDICTOR_A: DPCM A predictor 0: predictor1 1: predictor2

29.16.60 CSI_DPCM_CTRL_B_0

Offset: 0x28c | Byte Offset: 0xa30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_B: DPCM A compression ratio 0: BYPASS 1: 10_8_10 2: 10_7_10 3: 10_6_10 4: 12_8_12 5: 12_7_12 6: 12_6_12 7: 14_10_14 8: 14_8_14
0	0x0	DPCM_PREDICTOR_B: DPCM A predictor 0: predictor1 1: predictor2

29.16.61 CSI_STALL_COUNTER_0

Offset: 0x291 | Byte Offset: 0xa44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	STALL_SENSOR_B_COUNT
7:0	0x0	STALL_SENSOR_A_COUNT: Number of cycles to stall sensor A after every EOF

29.16.62 CSI_CSI_READONLY_STATUS_0

CSI Read Only Status

This register is used to return CSI read only status.

Offset: 0x292 | Byte Offset: 0xa48 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	CSI_PPB_ACTIVE: One only when Pixel Parser B is capturing frame data.
0	X	CSI_PPA_ACTIVE: One only when Pixel Parser A is capturing frame data.

29.16.63 CSI_CSI_SW_STATUS_RESET_0

Offset: 0x293 | Byte Offset: 0xa4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_STATUS_RESET: Reset CSI status and dbgcnt registers

29.16.64 CSI_CLKEN_OVERRIDE_0

Second-level clock enable override register

This can override the 2nd level clock enables in case of malfunction. Only exposed to software when needed.

Offset: 0x294 | Byte Offset: 0xa50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00xx00xxx00xxx00x00)

Bit	Reset	Description
18	CLK_GATED	CSI_CILB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
17	CLK_GATED	CSI_CILA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
14	CLK_GATED	CSI_PPB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
13	CLK_GATED	CSI_PPA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
9	CLK_GATED	CSI_HPB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
8	CLK_GATED	CSI_HPA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
4	CLK_GATED	CSI_FB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
3	CLK_GATED	CSI_FA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
1	CLK_GATED	CSI_DBG_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
0	CLK_GATED	CSI_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON

29.16.65 CSI_DEBUG_CONTROL_0

Debug Control

Offset: 0x295 | Byte Offset: 0xa54 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	R/W	Reset	Description
31	RO	X	DBG_CNT_ROLLED_2: Set when dbg_cnt_2 is incremented past max count, cleared when clr_dbg_cnt_2 is written with a value of 1.

Bit	R/W	Reset	Description
30:24	RW	X	DBG_CNT_SEL_2: Debug Count Select 2, this field selects what will be counted by debug counter 2. Encodings 00 to 15 select the set signal for one of the CSI_PIXEL_PARSER_A_STATUS register bits. In this case the select encoding of this field is the same as the bit position, in CSI_PIXEL_PARSER_A_STATUS, of the status bit whose set signal will be used to increment DBG_CNT_0. Encodings 16 to 31 select CSI_PIXEL_PARSER_B_STATUS. Encodings 32 to 39 select the set signal for one of the CSI_CIL_A_STATUS status bits. The least significant 3 bits of this select field give the bit position, in CSI_CIL_STATUS, of the status bit whose set signal pulses will be counted by dbg_cnt_0. Encodings 40 to 47 select CSI_CIL_B_STATUS. Encodings 48 to 55 select CSI_CIL_C_STATUS. Encodings 56 to 63 select CSI_CIL_D_STATUS. Encodings 96 to 103 selects CSI_CIL_E_STATUS. Other selections are given below: 64 - PPA Line packets processed. 65 - PPA short packets processed. 66 - Total packets processed by PPA. 67 - PPA Frame Starts Outputted. 68 - PPA Frame Ends Outputted. 69 - HPA Headers Parsed. 70 - HPA Headers Parsed with no ECC Errors. 71 to 79 - Reserved encoding. 80 - PPB Line packets processed. 81 - PPB short packets processed. 82 - Total packets processed by PPB. 83 - PPB Frame Starts Outputted. 84 - PPB Frame Ends Outputted. 85 - HPB Headers Parsed 86 - HPB Headers Parsed with no ECC Errors
23	RO	X	DBG_CNT_ROLLED_1: Set when dbg_cnt_1 is incremented past max count, cleared when clr_dbg_cnt_1 is written with a value of 1.
22:16	RW	X	DBG_CNT_SEL_1: Debug Count Select 1, this field selects what will be counted by debug counter 1. Encodings 00 to 15 select the set signal for one of the CSI_PIXEL_PARSER_A_STATUS register bits. In this case the select encoding of this field is the same as the bit position, in CSI_PIXEL_PARSER_A_STATUS, of the status bit whose set signal will be used to increment DBG_CNT_0. Encodings 16 to 31 select CSI_PIXEL_PARSER_B_STATUS. Encodings 32 to 39 select the set signal for one of the CSI_CIL_A_STATUS status bits. The least significant 3 bits of this select field give the bit position, in CSI_CIL_A_STATUS, of the status bit whose set signal pulses will be counted by dbg_cnt_0. Encodings 40 to 47 selects CSI_CIL_B_STATUS. Encodings 48 to 55 select CSI_CIL_C_STATUS. Encodings 56 to 63 select CSI_CIL_D_STATUS. Encodings 96 to 103 select CSI_CIL_E_STATUS. Other selections are given below: 64 - PPA Line packets processed. 65 - PPA short packets processed. 66 - Total packets processed by PPA. 67 - PPA Frame Starts Outputted. 68 - PPA Frame Ends Outputted. 69 - HPA Headers Parsed. 70 - HPA Headers Parsed with no ECC Errors. 71 to 79 - Reserved encoding. 80 - PPB Line packets processed. 81 - PPB short packets processed. 82 - Total packets processed by PPB. 83 - PPB Frame Starts Outputted. 84 - PPB Frame Ends Outputted. 85 - HPB Headers Parsed 86 - HPB Headers Parsed with no ECC Errors.
15	RO	X	DBG_CNT_ROLLED_0: Set when dbg_cnt_0 is incremented past max count, cleared when clr_dbg_cnt_0 is written with a value of 1.
14:8	RW	X	DBG_CNT_SEL_0: Debug Count Select 0, this field selects what will be counted by debug counter 0. Encodings 00 to 15 select the set signal for one of the CSI_PIXEL_PARSER_A_STATUS register bits. In this case the select encoding of this field is the same as the bit position, in CSI_PIXEL_PARSER_A_STATUS, of the status bit whose set signal will be used to increment DBG_CNT_0. Encodings 16 to 31 select CSI_PIXEL_PARSER_B_STATUS. Encodings 32 to 39 select the set signal for one of the CSI_CIL_A_STATUS status bits. The least significant 3 bits of this select field give the bit position, in CSI_CIL_A_STATUS, of the status bit whose set signal pulses will be counted by dbg_cnt_0. Encodings 40 to 47 selects CSI_CIL_B_STATUS. Encodings 48 to 55 select CSI_CIL_C_STATUS. Encodings 56 to 63 select CSI_CIL_D_STATUS. Encodings 96 to 103 select CSI_CIL_E_STATUS. Other selections are given below: 64 - PPA Line packets processed. 65 - PPA short packets processed. 66 - Total packets processed by PPA. 67 - PPA Frame Starts Outputted. 68 - PPA Frame Ends Outputted. 69 - HPA Headers Parsed. 70 - HPA Headers Parsed with no ECC Errors. 71 to 79 - Reserved encoding. 80 - PPB Line packets processed. 81 - PPB short packets processed. 82 - Total packets processed by PPB. 83 - PPB Frame Starts Outputted. 84 - PPB Frame Ends Outputted. 85 - HPB Headers Parsed 86 - HPB Headers Parsed with no ECC Errors.
7	RO	X	CLR_DBG_CNT_2: Clear Debug Counter 2, write a one to this bit to clear debug counter 2 and dbg_cnt_rolled_2.
6	RO	X	CLR_DBG_CNT_1: Clear Debug Counter 1, write a one to this bit to clear debug counter 1 and dbg_cnt_rolled_1.
5	RO	X	CLR_DBG_CNT_0: Clear Debug Counter 0, write a one to this bit to clear debug counter 0 and dbg_cnt_rolled_0.

Bit	R/W	Reset	Description
2	RO	X	CSIB_DBG_SF: When CSI-B is operating in a "Header Not Sent mode", writing a 1 to this bit indicates start frame (SF) or end frame (EF) control code. After the pixel parser is enabled, writing a 1 to this bit will start frame capture and send start frame (SF) control code. Writing a 1 to this bit again will stop frame capture and send end frame (EF) control code. "Header Not Sent mode" can be used as a debug mode to capture what the sensor is sending without interpreting the packets. Writing a 1 to this bit continually will generate SF and EF control codes. Note that a wait for MISC_CSI_PPB_FRAME_END syncpt is needed between an EF trigger for the current frame and an SF trigger for the next frame.
1	RO	X	CSIA_DBG_SF: When CSI-A is operating in a "Header Not Sent mode", writing a 1 to this bit indicates start frame (SF) or end frame (EF) control code. After the pixel parser is enabled, writing a 1 to this bit will start frame capture and send start frame (SF) control code. Writing a 1 to this bit again will stop frame capture and send end frame (EF) control code. "Header Not Sent mode" can be used as a debug mode to capture what the sensor is sending without interpreting the packets. Writing a 1 to this bit continually will generate SF and EF control codes. Note that a wait for MISC_CSI_PPA_FRAME_END syncpt is needed between an EF trigger for the current frame and an SF trigger for the next frame.
0	RW	0x0	DEBUG_EN: Debug Enable Second level CSI Debug clock is enabled. Debug counters 2, 1 & 0 are powered up. 0 = DISABLED : Debug counters 2, 1, and 0 are powered down. Second level CSI Debug clock is disabled. 1 = ENABLED

29.16.66 CSI_DEBUG_COUNTER_0_0

Debug Counter 0

This register can be used to count error conditions or packets processed.

Offset: 0x296 | Byte Offset: 0xa58 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DBG_CNT_0: When read returns the value of debug counter 0.

29.16.67 CSI_DEBUG_COUNTER_1_0

Debug Counter 1

This register can be used to count error conditions or packets processed.

Offset: 0x297 | Byte Offset: 0xa5c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DBG_CNT_1: When read returns the value of debug counter 1.

29.16.68 CSI_DEBUG_COUNTER_2_0

Debug Counter 2

This register can be used to count error conditions or packets processed.

Offset: 0x298 | Byte Offset: 0xa60 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DBG_CNT_2: When read returns the value of debug counter 2

29.16.69 CSI1_CSI_CAP_CIL_0

CIL Capability

Interface packets

Offset: 0x402 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:4	X	CIL_B_NUMLANES: CIL-B logic exists only if the returned value is other than _NONE. Returned value indicates how many lanes that CSI has. 0 = NONE 1 = ONE 2 = TWO
2:0	X	CIL_A_NUMLANES: CIL-A logic exists only if the returned value is other than _NONE. Returned value indicates how many lanes that CSI has. 0 = NONE 1 = ONE 2 = TWO

29.16.70 CSI1_CSI_CAP_CSI_0

CSI Capability

Offset: 0x406 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:4	X	CSIB_EXISTS: CSIB logic exists only if the returned value is other than _NO. 0 = NO 1 = YES
3:0	X	CSIA_EXISTS: CSIA logic exists only if the returned value is other than _NO. 0 = NO 1 = YES

29.16.71 CSI1_CSI_CAP_PP_0

PP Capabilities

Offset: 0x40a | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:4	X	PPB_EXISTS: PPB logic exists only if the returned value is other than _NO. 0 = NO 1 = YES
3:0	X	PPA_EXISTS: PPA logic exists only if the returned value is other than _NO. 0 = NO 1 = YES

29.16.72 CSI1_INPUT_STREAM_A_CONTROL_0

CSI Input Stream A Control

CSI PPA group

Offset: 0x40e | Byte Offset: 0x1038 | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_A_SKIP_PACKET_THRESHOLD: CSI-A Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_A_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE: Skip packet feature is enabled.
2	0x1	CSI_A_BYPASS_ALIGN: Bypass aligning CSIA and CSIB lanes if valids for the lanes are out of sync by a cycle.

Bit	Reset	Description
1:0	0x0	CSI_A_DATA_LANE: CSI-A Data Lane 0= 1 data lane 1= 2 data lanes 2= 3 data lanes 3= 4 data lanes Note: 3 data lanes is not supported in Test Pattern Generation mode

29.16.73 CSI1_PIXEL_STREAM_A_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x40f | Byte Offset: 0x103c | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

Bit	Reset	Description
29:28	X	CSI_PPA_PAD_FRAME: CSI Pixel Parser A Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPA_EXP_FRAME_HEIGHT. To do padding, the value in CSI_PPA_WORD_COUNT needs to be set to the number of input bytes in each line's payload. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.
27	X	CSI_PPA_HEADER_EC_ENABLE: CSI Pixel Parser A Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected or not. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser A. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB).
25:24	X	CSI_PPA_PAD_SHORT_LINE: CSI Pixel Parser A Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count). 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receive CSI output stream to hang up.
21:20	X	CSI_PPA_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser A Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPA_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by other CSI pixel stream processor. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED: output embedded data as 8-bpp arbitrary data stream.
19:16	X	CSI_PPA_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser A Output Format Options. This parameter specifies options for output data format. 0 = ARBITRARY: Output as 8-bit arbitrary data stream. This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE: Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed.
8	0x1	CSI_PPA_WC_CHECK: CSI Pixel Parser A Data WC Check. This parameter specifies whether the wordcount is checked. 0 = DISABLE: Word count Check is disabled 1 = ENABLE: Word count Check is enabled.
7	X	CSI_PPA_CRC_CHECK: CSI Pixel Parser A Data CRC Check. This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.

Bit	Reset	Description
6	X	CSI_PPA_WORD_COUNT_SELECT: CSI Pixel Parser A Word Count Select. This parameter is effective only if the packet header is sent as part of the stream. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPA_WORD_COUNT is different than the count in the packet header. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. It is recommended to always program this bit to HEADER.
5	X	CSI_PPA_DATA_IDENTIFIER: CSI Pixel Parser A Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPA_DATA_TYPE and against CSI_PPA_VIRTUAL_CHANNEL_ID). In this case, CSI_PPA_DATA_TYPE specifies the stream data format. 1 = ENABLED: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPA_DATA_TYPE and the CSI_PPA_VIRTUAL_CHANNEL_ID.
4	X	CSI_PPA_PACKET_HEADER: CSI Pixel Parser A Packet Header processing. This specifies whether packet header is sent in the beginning of the packet or not. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPA_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.
2:0	0x0	CSI_PPA_STREAM_SOURCE: CSI Pixel Parser A Stream Source 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B

29.16.74 CSI1_PIXEL_STREAM_A_CONTROL1_0

CSI Pixel Stream A Control 1

Offset: 0x410 | Byte Offset: 0x1040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	CSI_PPA_TOP_FIELD_FRAME_MASK: CSI Pixel Parser A Top Field Frame Mask
3:0	0x0	CSI_PPA_TOP_FIELD_FRAME: CSI Pixel Parser A Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise OR of ~(CSI_PPA_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPA_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet. Due to the reliance on WC field in packet header, interlaced support will not work if WC is selected from REGISTER mode.

29.16.75 CSI1_PIXEL_STREAM_A_GAP_0

CSI Pixel Stream A Gap

Offset: 0x411 | Byte Offset: 0x1044 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPA_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPA_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

29.16.76 CSI1_PIXEL_STREAM_PPA_COMMAND_0

CSI Pixel Parser A Command

Offset: 0x412 | Byte Offset: 0x1048 | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPA_START_MARKER_FRAME_MAX: CSI Pixel Parser A Start Marker Maximum. Start Frame is indicated when Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.
11:8	X	CSI_PPA_START_MARKER_FRAME_MIN: CSI Pixel Parser A Start Marker Minimum. Start Frame is indicated when Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.
4	X	CSI_PPA_VSYNC_START_MARKER: CSI Pixel Parser A VSYNC Start Marker 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPA_START_MARKER_FRAME_MIN and less than or equal to CSI_PPA_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC: Start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC.
2	0x1	CSI_PPA_SINGLE_SHOT: CSI Pixel Parser A Single Shot Mode. SW should clear this bit along with disabling the CSI_PPA_ENABLE, once a frame is captured. 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPA_ENABLE: CSI Pixel Parser A Enable. This parameter controls CSI Pixel Parser A to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST: reset (disable immediately). Enabling the pixel parser does not enable the corresponding input source to receive data. If pixel parser is enabled later than the corresponding input source, the CSI will keep on rejecting incoming stream, until it encounters a valid SF.

29.16.77 CSI1_PIXEL_STREAM_A_EXPECTED_FRAME_0

CSI Pixel Stream A Expected Frame

Offset: 0x413 | Byte Offset: 0x104c | Read/Write: R/W | Reset: 0x0000XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:4	X	PPA_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	0x0	PPA_ENABLE_LINE_TIMEOUT: When set to one enables checking of the time between start line requests from the Header Parser to CSI-PPA. A fake EF will be outputted by CSI-PPA if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines outputted, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPA_PAD_FRAME.

29.16.78 CSI1_CSI_PIXEL_PARSER_A_INTERRUPT_MASK_0

CSI Pixel Parser A Interrupt Mask

Offset: 0x414 | Byte Offset: 0x1050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPA_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPA_UNC_HDR_ERR. 0 = DISABLED: Don't generate an interrupt when HPA_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPA_UNC_HDR_ERR is set.
10	0x0	PPA_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPA_SPARE_STATUS_1. 0 = DISABLED: Don't generate an interrupt when PPA_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPA_SPARE_STATUS_1 is set.
9	0x0	PPA_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPA_INTERFRAME_LINE. 0 = DISABLED: Don't generate an interrupt when PPA_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPA_INTERFRAME_LINE is set.

Bit	Reset	Description
8	0x0	PPA_EXTRA_SF_INT_MASK: Interrupt Mask for PPA_EXTRA_SF. 0 = DISABLED: Don't generate an interrupt when PPA_EXTRA_SF is set. 1 = ENABLED: Generate an interrupt when PPA_EXTRA_SF is set.
7	0x0	PPA_SHORT_FRAME_INT_MASK: Interrupt Mask for PPA_SHORT_FRAME. 0 = DISABLED: Don't generate an interrupt when PPA_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPA_SHORT_FRAME is set.
6	0x0	PPA_STMERR_INT_MASK: Interrupt Mask for PPA_STMERR. 0 = DISABLED: Don't generate an interrupt when PPA_STMERR is set. 1 = ENABLED: Generate an interrupt when PPA_STMERR is set.
5	0x0	PPA_FIFO_OVRF_INT_MASK: Interrupt Mask for PPA_FIFO_OVRF. 0 = DISABLED: Don't generate an interrupt when PPA_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPA_FIFO_OVRF is set.
4	0x0	PPA_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPA_PL_CRC_ERR. 0 = DISABLED: Don't generate an interrupt when PPA_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPA_PL_CRC_ERR is set.
3	0x0	PPA_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPA_SL_PKT_DROPPED. 0 = DISABLED: Don't generate an interrupt when PPA_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PKT_DROPPED is set.
2	0x0	PPA_SL_PROCESSED_INT_MASK: Interrupt Mask for PPA_SL_PROCESSED. 0 = DISABLED: Don't generate an interrupt when PPA_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PROCESSED is set.
1	0x0	PPA_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPA_ILL_WD_CNT. 0 = DISABLED: Don't generate an interrupt when PPA_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPA_ILL_WD_CNT is set.
0	0x0	PPA_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPA_HDR_ERR_COR. 0 = DISABLED: Don't generate an interrupt when PPA_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPA_HDR_ERR_COR is set.

29.16.79 CSI1_CSI_PIXEL_PARSER_A_STATUS_0

Pixel Parser A Status

Each status bit is cleared to zero when its bit position is written with one. For example, write 0x2 to CSI_PIXEL_PARSER_STATUS will clear only PPA_ILL_WD_CNT.

Offset: 0x415 | Byte Offset: 0x1054 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	HPA_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser A parses a header with a multi-bit error. This error will be detected by the headers ECC, but cannot be corrected. The packet will be discarded.
10	X	PPA_SPARE_STATUS_1: PPA Spare Status bit. This bit will get set when Pixel Parser A has a line timeout. Line timeout needs to be enabled by setting PPA_ENABLE_LINE_TIMEOUT and programming PPA_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPA_INTERFRAME_LINE: Set when CSI-PPA receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.
8	X	PPA_EXTRA_SF: Set when CSI-PPA receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPA will insert a fake EF and the drop the current frame with the Correct SF.
7	X	PPA_SHORT_FRAME: Set when CSI-PPA receives a short frame. This bit gets set even if CSI_PPA_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPA_STMERR: Stream Error, set when the control output of PPA does not follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
5	X	PPA_FIFO_OVRF: FIFO Overflow. Set when the FIFO that is feeding packets to PPA overflows.
4	X	PPA_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPA had a payload CRC error.
3	X	PPA_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_A_SKIP_PACKET_THRESHOLD when padding a short line.

Bit	Reset	Description
2	X	PPA_SL_PROCESSED: Short Line Processed. Set when a line with a payload that is shorter than its packet header word count processed by PPA.
1	X	PPA_ILL_WD_CNT: Illegal Word Count. Set when a line with a word count that does not generate an integer number of pixels (unused bytes at the end of payload) is processed by PPA.
0	X	PPA_HDR_ERR_COR: Header Error Corrected. Set when a packet that was processed by PPA has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_A_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

29.16.80 CSI1_CSI_SW_SENSOR_A_RESET_0

Offset: 0x416 | Byte Offset: 0x1058 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_A_RESET: Reset CSI sensor A

29.16.81 CSI1_INPUT_STREAM_B_CONTROL_0

CSI Input Stream B Control

CSI PPB group

Offset: 0x41b | Byte Offset: 0x106c | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_B_SKIP_PACKET_THRESHOLD: CSI-B Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_B_SKIP_PACKET_THRESHOLD_ENABLE: 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled.
2	0x1	CSI_B_BYPASS_ALIGN: Bypass aligning CSIB lanes if valids for the lanes are out of sync by a cycle
1:0	0x0	CSI_B_DATA_LANE: CSI-B Data Lane 0= 1 data lane 1= 2 data lanes 2= 3 data lanes (not supported) 3= 4 data lanes (not supported) Note: 3 data lanes is not supported in Test Pattern Generation mode

29.16.82 CSI1_PIXEL_STREAM_B_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x41c | Byte Offset: 0x1070 | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

Bit	Reset	Description
29:28	X	CSI_PPB_PAD_FRAME: CSI Pixel Parser B Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPB_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPB_WORD_COUNT needs to be set to the number of input bytes in each lines payload. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.

Bit	Reset	Description
27	X	CSI_PPB_HEADER_EC_ENABLE: CSI Pixel Parser B Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected, or not. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser B. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB).
25:24	X	CSI_PPB_PAD_SHORT_LINE: CSI Pixel Parser B Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count) 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receive CSI output stream to hang up.
21:20	X	CSI_PPB_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser B Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPB_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by another CSI pixel stream processor. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED: output embedded data as 8-bpp arbitrary data stream
19:16	X	CSI_PPB_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser B Output Format Options. This parameter specifies output data format. 0 = ARBITRARY: Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE: Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed.
8	0x1	CSI_PPB_WC_CHECK: CSI Pixel Parser B Data WC Check. This parameter specifies whether the wordcount is checked. 0 = DISABLE: Word count Check is disabled 1 = ENABLE: Word count Check is enabled.
7	X	CSI_PPB_CRC_CHECK: CSI Pixel Parser B Data CRC Check. This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.
6	X	CSI_PPB_WORD_COUNT_SELECT: CSI Pixel Parser B Word Count Select. This parameter is effective only if the packet header is sent as part of the stream. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPB_WORD_COUNT is different than the count in the packet header. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. It is recommended to always program this bit to HEADER.
5	X	CSI_PPB_DATA_IDENTIFIER: CSI Pixel Parser B Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPB_DATA_TYPE and against CSI_PPB_VIRTUAL_CHANNEL_ID). In this case, CSI_PPB_DATA_TYPE specifies the stream data format. 1 = ENABLED: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPB_DATA_TYPE and the CSI_PPB_VIRTUAL_CHANNEL_ID.
4	X	CSI_PPB_PACKET_HEADER: CSI Pixel Parser B Packet Header processing. This specifies whether packet header is sent in the beginning of packet or not. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPB_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.

Bit	Reset	Description
2:0	0x0	CSI_PPB_STREAM_SOURCE: CSI Pixel Parser B Stream Source CSI Interface B 0 = CSI_A: CSI Interface A 1 = CSI_B

29.16.83 CSI1_PIXEL_STREAM_B_CONTROL1_0

CSI Pixel Stream B Control 1

Offset: 0x41d | Byte Offset: 0x1074 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	CSI_PPB_TOP_FIELD_FRAME_MASK: CSI Pixel Parser B Top Field Frame Mask
3:0	0x0	CSI_PPB_TOP_FIELD_FRAME: CSI Pixel Parser B Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise OR of ~(CSI_PPB_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPB_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet. Due to the reliance on WC field in packet header, interlaced support will not work if WC is selected from REGISTER mode.

29.16.84 CSI1_PIXEL_STREAM_B_GAP_0

CSI Pixel Stream B Gap

Offset: 0x41e | Byte Offset: 0x1078 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPB_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPB_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

29.16.85 CSI1_PIXEL_STREAM_PPB_COMMAND_0

CSI Pixel Parser B Command

Offset: 0x41f | Byte Offset: 0x107c | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPB_START_MARKER_FRAME_MAX: CSI Pixel Parser B Start Marker Maximum. Start Frame is indicated when Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.
11:8	X	CSI_PPB_START_MARKER_FRAME_MIN: CSI Pixel Parser B Start Marker Minimum. Start Frame is indicated when Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.
4	X	CSI_PPB_VSYNC_START_MARKER: CSI Pixel Parser B VSYNC Start Marker. 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPB_START_MARKER_FRAME_MIN and less than or equal to CSI_PPB_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC: Start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC.
2	0x1	CSI_PPB_SINGLE_SHOT: CSI Pixel Parser B Single Shot Mode. Software should clear it along with disabling the CSI_PPB_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1:0	0x0	CSI_PPB_ENABLE: CSI Pixel Parser B Enable. This parameter controls CSI Pixel Parser B to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST: reset (disable immediately). Enabling the pixel parser does not enable the corresponding input source to receive data. If pixel parser is enabled later than the corresponding input source, the CSI will keep on rejecting incoming stream, until it encounters a valid SF.

29.16.86 CSI1_PIXEL_STREAM_B_EXPECTED_FRAME_0

CSI Pixel Stream B Expected Frame

Offset: 0x420 | Byte Offset: 0x1080 | Read/Write: R/W | Reset: 0x0000XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:4	X	PPB_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	0x0	PPB_ENABLE_LINE_TIMEOUT: When set to one enables checking of the time between start line requests from the Header Parser to CSI-PPB. A fake EF will be outputted by CSI-PPB if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines outputted, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPB_PAD_FRAME.

29.16.87 CSI1_CSI_PIXEL_PARSER_B_INTERRUPT_MASK_0

CSI Pixel Parser B Interrupt Mask

Offset: 0x421 | Byte Offset: 0x1084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPB_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPB_UNC_HDR_ERR. 0 = DISABLED: Don't generate an interrupt when HPB_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPB_UNC_HDR_ERR is set.
10	0x0	PPB_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPB_SPARE_STATUS_1. 0 = DISABLED: Don't generate an interrupt when PPB_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPB_SPARE_STATUS_1 is set.
9	0x0	PPB_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPB_INTERFRAME_LINE. 0 = DISABLED: Don't generate an interrupt when PPB_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPB_INTERFRAME_LINE is set.
8	0x0	PPB_EXTRA_SF_INT_MASK: Interrupt Mask for PPB_EXTRA_SF. 0 = DISABLED: Don't generate an interrupt when PPB_EXTRA_SF is set. 1 = ENABLED: Generate an interrupt when PPB_EXTRA_SF is set.
7	0x0	PPB_SHORT_FRAME_INT_MASK: Interrupt Mask for PPB_SHORT_FRAME. 0 = DISABLED: Don't generate an interrupt when PPB_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPB_SHORT_FRAME is set.
6	0x0	PPB_STMERR_INT_MASK: Interrupt Mask for PPB_STMERR. 0 = DISABLED: Don't generate an interrupt when PPB_STMERR is set. 1 = ENABLED: Generate an interrupt when PPB_STMERR is set.
5	0x0	PPB_FIFO_OVRF_INT_MASK: Interrupt Mask for PPB_FIFO_OVRF. 0 = DISABLED: Don't generate an interrupt when PPB_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPB_FIFO_OVRF is set.
4	0x0	PPB_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPB_PL_CRC_ERR. 0 = DISABLED: Don't generate an interrupt when PPB_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPB_PL_CRC_ERR is set.
3	0x0	PPB_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPB_SL_PKT_DROPPED. 0 = DISABLED: Don't generate an interrupt when PPB_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PKT_DROPPED is set.
2	0x0	PPB_SL_PROCESSED_INT_MASK: Interrupt Mask for PPB_SL_PROCESSED. 0 = DISABLED: Don't generate an interrupt when PPB_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PROCESSED is set.

Bit	Reset	Description
1	0x0	PPB_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPB_ILL_WD_CNT. 0 = DISABLED: Don't generate an interrupt when PPB_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPB_ILL_WD_CNT is set.
0	0x0	PPB_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPB_HDR_ERR_COR. 0 = DISABLED: Don't generate an interrupt when PPB_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPB_HDR_ERR_COR is set.

29.16.88 CSI1_CSI_PIXEL_PARSER_B_STATUS_0

Pixel Parser B Status

Offset: 0x422 | Byte Offset: 0x1088 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	HPB_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser B parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected.
10	X	PPB_SPARE_STATUS_1: PPB Spare Status bit. This bit will get set when Pixel Parser B has a line timeout. Line timeout needs to be enabled by setting PPB_ENABLE_LINE_TIMEOUT and programming PPB_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPB_INTERFRAME_LINE: Set when CSI-PPB receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.
8	X	PPB_EXTRA_SF: Set when CSI-PPB receives a SF when it is expecting an EF. This happens when the EF of the frame gets corrupted before arriving to the CSI. CSI-PPB will insert a fake EF and drop the current frame with the Correct SF.
7	X	PPB_SHORT_FRAME: Set when CSI-PPB receives a short frame. This bit gets set even if CSI_PPB_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPB_STMERR: Stream Error. Set when the control output of PPB doesn't follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
5	X	PPB_FIFO_OVRF: FIFO Overflow. Set when the FIFO that is feeding packets to PPB overflows.
4	X	PPB_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPB had a payload CRC error.
3	X	PPB_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_B_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPB_SL_PROCESSED: Short Line Processed. Set when a line with a payload that is shorter than its packet header word count is processed by PPB.
1	X	PPB_ILL_WD_CNT: Illegal Word Count. Set when a line with a word count that doesn't generate an integer number of pixels (unused bytes at the end of payload) is processed by PPB.
0	X	PPB_HDR_ERR_COR: Header Error Corrected. Set when a packet that was processed by PPB has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_B_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

29.16.89 CSI1_CSI_SW_SENSOR_B_RESET_0

Offset: 0x423 | Byte Offset: 0x108c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_B_RESET: Reset CSI sensor B

29.16.90 CSI1_PHY_CIL_COMMAND_0

CSI Phy and CIL Command

CSI CIL group

Offset: 0x442 | Byte Offset: 0x1108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00xxxxxx00)

Bit	Reset	Description
9:8	0x0	CSI_B_PHY_CIL_ENABLE: CSI B Phy and CIL Enable. This parameter controls CSI B Phy and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: disable (reset)
1:0	0x0	CSI_A_PHY_CIL_ENABLE: CSI A Phy and CIL Enable. This parameter controls CSI A Phy and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: disable (reset)

29.16.91 CSI1_CIL_PAD_CONFIG0_0

CIL Pad Configuration 0

Offset: 0x443 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000xxxxxxxx)

Bit	Reset	Description
15:8	0x0	PAD_CIL_SPARE: Spare bit for CIL BIAS Config. PAD_CIL_SPARE[7] is used to flush VI's Y-FIFO when it is being used as a stream source for one of the Pixel Parsers. Setting PAD_CIL_SPARE[7] to 1 will hold vi2csi_host_stall low, which will force VI's Y-FIFO to be purged. PAD_CIL_SPARE[7] must be low for the pixel parser to receive source data from VI's Y-FIFO.

29.16.92 CSI1_CILA_PAD_CONFIG0_0

CIL-A Pad Configuration 0

CSI CILA group

Offset: 0x44b | Byte Offset: 0x112c | Read/Write: R/W | Reset: 0x00010007 (0b0000x000000000010000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILA_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. Will cause stress and need waiver to use.
30:28	0x0	PAD_CILA_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only.
26:24	0x0	PAD_CILA_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only.
23:21	0x0	PAD_CILA_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only.
20:18	0x0	PAD_CILA_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only
17:16	0x1	PAD_AB_BK_MODE: 00: Two independent 2x bricks. 01: one 4x brick, received clock from partition A is used. Clock from partition B is not used. 10: one 4x brick, received clock from partition B is used. Clock from partition A is not used. 11: illegal
15	0x0	PAD_CILA_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILA_INADJ1: Bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILA_INADJ0: Bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILA_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILA_PDVCLAMP: Power down regulator which supplies current to serializer de-serializer logic
2	0x1	PAD_CILA_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILA_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

29.16.93 CSI1_CILA_PAD_CONFIG1_0

CIL-A Pad Configuration 4

Offset: 0x44c | Byte Offset: 0x1130 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILA_SPARE_RO: Spare Read only bits for CILA Config
15:8	RW	0x0	PAD_CILA_SPARE: Spare bits for CILA Config. PAD_CILA_SPARE[15] is being used to disable the CSI-A RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled, 1: push blocking enabled
7:6	RW	0x0	PAD_CILA_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only.
5:4	RW	0x0	PAD_CILA_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only.
3:2	RW	0x0	PAD_CILA_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only.
1:0	RW	0x0	PAD_CILA_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis. 11=max. This control is unused for CSI MIPI pads, which are receive only.

29.16.94 CSI1_PHY_CILA_CONTROL0_0

CSI-A PHY and CIL Control

Offset: 0x44d | Byte Offset: 0x1134 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILA_CLK_SETTLE: settle time for clk lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz lp clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 csicil cycles for the default value. So the programming value (0 to 15) of this field works like this: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) . . 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) . 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILA_BYPASS_LP_SEQ: The LP signals should sequence through LP11->LP01->LP00 states, to indicate to CLOCK CIL about the mode switching to HS Rx mode. In case the camera is enabled earlier than CIL , it is highly likely that the camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
5:0	0x2	CILA_THS_SETTLE: settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field. 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

29.16.95 CSI1_CSI_CIL_A_INTERRUPT_MASK_0

CSI Control and Interface Logic A Interrupt Mask

Offset: 0x44e | Byte Offset: 0x1138 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
9	0x0	CILA_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CLK_CTRL_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_CLK_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CLK_CTRL_ERR is set.
6	0x0	CILA_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILA_ESC_DATA_REC. 0 = DISABLED: Don't generate an interrupt when CILA_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_DATA_REC is set.

Bit	Reset	Description
5	0x0	CILA_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILA_ESC_CMD_REC. 0 = DISABLED: Don't generate an interrupt when CILA_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_CMD_REC is set.
4	0x0	CILA_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CTRL_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CTRL_ERR is set.
2	0x0	CILA_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILA_SYNC_ESC_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SYNC_ESC_ERR is set.
1	0x0	CILA_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_MB_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_MB_ERR is set.
0	0x0	CILA_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_SB_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_SB_ERR is set.

29.16.96 CSI1_CSI_CIL_A_STATUS_0

CSI Control and Interface Logic A Status

Each status bit is cleared to zero when its bit position is written with one. For example writing 0x2 to CSI_CIL_A_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 0x44f | Byte Offset: 0x113c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILA_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-A receives an Escape Mode Data byte. The Data Byte can be read from bits 7-0 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILA_ESC_CMD_REC is set.
5	X	CILA_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-A receives an Escape Mode Command byte. The Command Byte can be read from bits 7-0 of ESCAPE_MODE_COMMAND.
4	X	CILA_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILA_SYNC_ESC_ERR: Sync Escape Error. Set when CIL-A detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
1	X	CILA_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packets SOT bytes. The packet will be discarded.
0	X	CILA_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packet’s Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

29.16.97 CSI1_CSI_CILA_STATUS_0

CSI-CILA Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x450 | Byte Offset: 0x1140 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILA_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILA_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-1. The packet will be discarded.
16	X	CILA_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-A for processing.
6	X	CILA_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning

Bit	Reset	Description
5	X	CILA_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-0. The packet will be discarded.
4	X	CILA_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-A for processing.
0	X	CILA_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-A detects incorrect line state sequence on clk lane

29.16.98 CSI1_CIL_A_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-A.

Offset: 0x451 | Byte Offset: 0x1144 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILA_ESC_CMD_BYTE: CIL-A Escape Mode Command Byte. This is the 8 bit entry command that was received, by CIL-A, during the last escape Mode sequence. CIL-A monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILA_ESC_CMD_REC status bit is set.

29.16.99 CSI1_CIL_A_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-A.

Offset: 0x452 | Byte Offset: 0x1148 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILA_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte. When read this field returns the last Escape Mode Data byte that was received by CIL-A. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILA_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte comes in.

29.16.100 CSI1_CSICIL_SW_SENSOR_A_RESET_0

Offset: 0x453 | Byte Offset: 0x114c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_A_RESET: Reset CSICIL sensor A.

29.16.101 CSI1_CILB_PAD_CONFIG0_0

CIL-B Pad Configuration 0

CSI CILB group

Offset: 0x458 | Byte Offset: 0x1160 | Read/Write: R/W | Reset: 0x00000007 (0b0000x000000000xx0000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILB_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. will cause stress and need waiver to use
30:28	0x0	PAD_CILB_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only.

Bit	Reset	Description
26:24	0x0	PAD_CILB_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
23:21	0x0	PAD_CILB_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only
20:18	0x0	PAD_CILB_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only
15	0x0	PAD_CILB_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILB_INADJ1: bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILB_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILB_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILB_PDVCLAMP: Power down regulator which supplies current to serializer/de-serializer logic
2	0x1	PAD_CILB_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILB_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

29.16.102 CSI1_CILB_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 0x459 | Byte Offset: 0x1164 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILB_SPARE_RO: Spare Read only bits for CILB Config
15:8	RW	0x0	PAD_CILB_SPARE: Spare bits for CILB Config. PAD_CILB_SPARE[15] is being used to disable the CSI-B RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled, 1: push blocking enabled
7:6	RW	0x0	PAD_CILB_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
5:4	RW	0x0	PAD_CILB_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
3:2	RW	0x0	PAD_CILB_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
1:0	RW	0x0	PAD_CILB_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only

29.16.103 CSI1_PHY_CILB_CONTROL0_0

CSI-B Phy and CIL Control

Offset: 0x45a | Byte Offset: 0x1168 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILB_CLK_SETTLE: settle time for clk lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 csicil cycles for default value. So the programming value (0 to 15) of this field works like this 0 = 15 cilclk cycles (default internal delay), 1 = 15 - 7 (8 cilclk cycles), 2 = 15 - 6 (9 cilclk cycles), . . 7 = 15 - 1 (14 cilclk cycles), 8 = 15 - 0 (default internal delay), 9 = 15 + 1 (16 cilclk cycles), . 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILB_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
5:0	0x2	CILB_THS_SETTLE: settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (102 MHz lp clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

29.16.104 CSI1_CSI_CIL_B_INTERRUPT_MASK_0

CSI Control and Interface Logic B Interrupt Mask

Offset: 0x45b | Byte Offset: 0x116c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
6	0x0	CILB_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILB_ESC_DATA_REC. 0 = DISABLED: Don't generate an interrupt when CILB_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_DATA_REC is set.
5	0x0	CILB_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILB_ESC_CMD_REC. 0 = DISABLED: Don't generate an interrupt when CILB_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_CMD_REC is set.
4	0x0	CILB_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CTRL_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_CTRL_ERR is set.
2	0x0	CILB_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILB_SYNC_ESC_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SYNC_ESC_ERR is set.
1	0x0	CILB_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_MB_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_MB_ERR is set.
0	0x0	CILB_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_SB_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_SB_ERR is set.

29.16.105 CSI1_CSI_CIL_B_STATUS_0

CSI Control and Interface Logic B Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_B_STATUS will clear only CILB_SOT_MB_ERR.

Offset: 0x45c | Byte Offset: 0x1170 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILB_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-B receives an Escape Mode Data byte. The Data Byte can be read from bits 23-16 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILB_ESC_CMD_REC is set.
5	X	CILB_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-B receives an Escape Mode Command byte. The Command Byte can be read from bits 23-16 of ESCAPE_MODE_COMMAND.
4	X	CILB_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILB_SYNC_ESC_ERR: Sync Escape Error. Set when CIL-B detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
1	X	CILB_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packets SOT bytes. The packet will be discarded.
0	X	CILB_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-B detects a single bit error in one of the packets start of transmission bytes. The packet will be sent to CSI-B for processing.

29.16.106 CSI1_CSI_CILB_STATUS_0

CSI-CILB Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x45d | Byte Offset: 0x1174 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILB_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning

Bit	Reset	Description
17	X	CILB_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-1. The packet will be discarded.
16	X	CILB_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-B for processing.
6	X	CILB_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILB_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-0. The packet will be discarded.
4	X	CILB_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-B detects a single bit error in one of the packets Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-B for processing.
0	X	CILB_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-B detects incorrect line state sequence on clk lane

29.16.107 CSI1_CIL_B_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-B.

Offset: 0x45e | Byte Offset: 0x1178 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILB_ESC_CMD_BYTE: CIL-B Escape Mode Command Byte. This is the 8 bit entry command that was received, by CIL-B, during the last escape Mode sequence. CIL-B monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILB_ESC_CMD_REC status bit is set.

29.16.108 CSI1_CIL_B_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-B.

Offset: 0x45f | Byte Offset: 0x117c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILB_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte. When read, this field returns the last Escape Mode Data byte that was received by CIL-B. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILB_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte comes in.

29.16.109 CSI1_CSICIL_SW_SENSOR_B_RESET_0

Offset: 0x460 | Byte Offset: 0x1180 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_B_RESET: Reset CSICIL sensor B

29.16.110 CSI1_PATTERN_GENERATOR_CTRL_A_0

CSI common group

Offset: 0x471 | Byte Offset: 0x11c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_A: Mode for Sensor A 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_A: Automatic phase increment mode for sensor A
0	0x0	PG_ENABLE_A: Enable Pattern Generator for sensor A

29.16.111 CSI1_PG_BLANK_A_0

Offset: 0x472 | Byte Offset: 0x11c8 | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_A: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_A: Horizontal Blanking for PG

29.16.112 CSI1_PG_PHASE_A_0

Offset: 0x473 | Byte Offset: 0x11cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_A: Initial Phase

29.16.113 CSI1_PG_RED_FREQ_A_0

Offset: 0x474 | Byte Offset: 0x11d0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.114 CSI1_PG_RED_FREQ_RATE_A_0

Offset: 0x475 | Byte Offset: 0x11d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.115 CSI1_PG_GREEN_FREQ_A_0

Offset: 0x476 | Byte Offset: 0x11d8 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.116 CSI1_PG_GREEN_FREQ_RATE_A_0

Offset: 0x477 | Byte Offset: 0x11dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.117 CSI1_PG_BLUE_FREQ_A_0

Offset: 0x478 | Byte Offset: 0x11e0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.118 CSI1_PG_BLUE_FREQ_RATE_A_0

Offset: 0x479 | Byte Offset: 0x11e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.119 CSI1_PG_AOHDR_A_0

Offset: 0x47a | Byte Offset: 0x11e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:1	0x0	PG_AOHDR_GAIN_RATIO_A: Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ration (short rows >> 2)
0	0x0	PG_AOHDR_ENABLE_A: AOHDR enable

29.16.120 CSI1_PATTERN_GENERATOR_CTRL_B_0

Offset: 0x47e | Byte Offset: 0x11f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_B: Mode for Sensor B 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_B: Automatic phase increment mode for sensor B
0	0x0	PG_ENABLE_B: Enable Pattern Generator for sensor B

29.16.121 CSI1_PG_BLANK_B_0

Offset: 0x47f | Byte Offset: 0x11fc | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_B: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_B: Horizontal Blanking for PG

29.16.122 CSI1_PG_PHASE_B_0

Offset: 0x480 | Byte Offset: 0x1200 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_B: Initial Phase

29.16.123 CSI1_PG_RED_FREQ_B_0

Offset: 0x481 | Byte Offset: 0x1204 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.124 CSI1_PG_RED_FREQ_RATE_B_0

Offset: 0x482 | Byte Offset: 0x1208 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.125 CSI1_PG_GREEN_FREQ_B_0

Offset: 0x483 | Byte Offset: 0x120c | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.126 CSI1_PG_GREEN_FREQ_RATE_B_0

Offset: 0x484 | Byte Offset: 0x1210 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.127 CSI1_PG_BLUE_FREQ_B_0

Offset: 0x485 | Byte Offset: 0x1214 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.128 CSI1_PG_BLUE_FREQ_RATE_B_0

Offset: 0x486 | Byte Offset: 0x1218 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.129 CSI1_PG_AOHDR_B_0

Offset: 0x487 | Byte Offset: 0x121c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:1	0x0	PG_AOHDR_GAIN_RATIO_B: Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ration (short rows >> 2)

Bit	Reset	Description
0	0x0	PG_AOHDR_ENABLE_B: AOHDR enable

29.16.130 CSI1_DPCM_CTRL_A_0

Offset: 0x48b | Byte Offset: 0x122c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_A: DPCM A compression ratio 0: BYPASS 1: 10_8_10 2: 10_7_10 3: 10_6_10 4: 12_8_12 5: 12_7_12 6: 12_6_12 7: 14_10_14 8: 14_8_14
0	0x0	DPCM_PREDICTOR_A: DPCM A predictor 0: predictor1, 1: predictor2

29.16.131 CSI1_DPCM_CTRL_B_0

Offset: 0x48c | Byte Offset: 0x1230 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_B: DPCM A compression ratio 0: BYPASS 1: 10_8_10 2: 10_7_10 3: 10_6_10 4: 12_8_12 5: 12_7_12 6: 12_6_12 7: 14_10_14 8: 14_8_14
0	0x0	DPCM_PREDICTOR_B: DPCM A predictor 0: predictor1, 1: predictor2

29.16.132 CSI1_STALL_COUNTER_0

Offset: 0x491 | Byte Offset: 0x1244 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	STALL_SENSOR_B_COUNT
7:0	0x0	STALL_SENSOR_A_COUNT: Number of cycles to stall sensor A after every EOF

29.16.133 CSI1_CSI_READONLY_STATUS_0

CSI Read Only Status, this register is used to return CSI read only status.

Offset: 0x492 | Byte Offset: 0x1248 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	CSI_PPB_ACTIVE: One only when Pixel Parser B is capturing frame data.
0	X	CSI_PPA_ACTIVE: One only when Pixel Parser A is capturing frame data.

29.16.134 CSI1_CSI_SW_STATUS_RESET_0

Offset: 0x493 | Byte Offset: 0x124c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_STATUS_RESET: Reset CSI status and dbgcnt registers

29.16.135 CSI2_CSI_CAP_CIL_0

CIL Capability

Interface packets

Offset: 0x602 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:4	X	CIL_B_NUMLANES: CIL-B logic exists only if the returned value is other than _NONE. Returned value indicates how many lanes that CSI has. 0 = NONE 1 = ONE 2 = TWO
2:0	X	CIL_A_NUMLANES: CIL-A logic exists only if the returned value is other than _NONE. Returned value indicates how many lanes that CSI has. 0 = NONE 1 = ONE 2 = TWO

29.16.136 CSI2_CSI_CAP_CSI_0

CSI Capability

Offset: 0x606 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:4	X	CSIB_EXISTS: CSIB logic exists only if the returned value is other than _NO. 0 = NO 1 = YES
3:0	X	CSIA_EXISTS: CSIA logic exists only if the returned value is other than _NO. 0 = NO 1 = YES

29.16.137 CSI2_CSI_CAP_PP_0

PP Capabilities

Offset: 0x60a | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:4	X	PPB_EXISTS: PPB logic exists only if the returned value is other than _NO. 0 = NO 1 = YES
3:0	X	PPA_EXISTS: PPA logic exists only if the returned value is other than _NO. 0 = NO 1 = YES

29.16.138 CSI2_INPUT_STREAM_A_CONTROL_0

CSI Input Stream A Control

CSI PPA group

Offset: 0x60e | Byte Offset: 0x1838 | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_A_SKIP_PACKET_THRESHOLD: CSI-A Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_A_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE: Skip packet feature is enabled.

Bit	Reset	Description
2	0x1	CSI_A_BYPASS_ALIGN: Bypass aligning CSIA and CSIB lanes if valids for the lanes are out of sync by a cycle
1:0	0x0	CSI_A_DATA_LANE: CSI-A Data Lane 0= 1 data lane 1= 2 data lanes 2= 3 data lanes 3= 4 data lanes Note: 3 data lanes is not supported in Test Pattern Generation mode

29.16.139 CSI2_PIXEL_STREAM_A_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x60f | Byte Offset: 0x183c | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

Bit	Reset	Description
29:28	X	CSI_PPA_PAD_FRAME: CSI Pixel Parser A Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPA_EXP_FRAME_HEIGHT. To do padding, the value in CSI_PPA_WORD_COUNT needs to be set to the number of input bytes in each line's payload. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.
27	X	CSI_PPA_HEADER_EC_ENABLE: CSI Pixel Parser A Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected, or not. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser A. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB).
25:24	X	CSI_PPA_PAD_SHORT_LINE: CSI Pixel Parser A Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count) 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receives CSI output stream to hang up.
21:20	X	CSI_PPA_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser A Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPA_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by other CSI pixel stream processor. 1 = EMBEDDED: output embedded data as 8-bpp arbitrary data stream. 0 = DISCARD: discard (throw away) embedded data
19:16	X	CSI_PPA_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser A Output Format Options. This parameter specifies options for output data format. 0 = ARBITRARY: Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE: Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed.
8	0x1	CSI_PPA_WC_CHECK: CSI Pixel Parser A Data WC Check. This parameter specifies whether the wordcount is checked. 0 = DISABLE: Word count Check is disabled 1 = ENABLE: Word count Check is enabled.

Bit	Reset	Description
7	X	CSI_PPA_CRC_CHECK: CSI Pixel Parser A Data CRC Check. This parameter specifies whether the last 2 bytes of the packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.
6	X	CSI_PPA_WORD_COUNT_SELECT: CSI Pixel Parser A Word Count Select. This parameter is effective only if packet header is sent as part of the stream. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPA_WORD_COUNT is different than the count in the packet header. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. It is recommended to always program this bit to HEADER.
5	X	CSI_PPA_DATA_IDENTIFIER: CSI Pixel Parser A Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPA_DATA_TYPE and against CSI_PPA_VIRTUAL_CHANNEL_ID). In this case, CSI_PPA_DATA_TYPE specifies the stream data format. 1 = ENABLED: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPA_DATA_TYPE and the CSI_PPA_VIRTUAL_CHANNEL_ID.
4	X	CSI_PPA_PACKET_HEADER: CSI Pixel Parser A Packet Header processing. This specifies whether packet header is sent in the beginning of packet or not. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPA_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.
2:0	0x0	CSI_PPA_STREAM_SOURCE: CSI Pixel Parser A Stream Source 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B

29.16.140 CSI2_PIXEL_STREAM_A_CONTROL1_0

CSI Pixel Stream A Control 1

Offset: 0x610 | Byte Offset: 0x1840 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	CSI_PPA_TOP_FIELD_FRAME_MASK: CSI Pixel Parser A Top Field Frame Mask
3:0	0x0	CSI_PPA_TOP_FIELD_FRAME: CSI Pixel Parser A Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise OR of ~(CSI_PPA_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPA_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet. Due to the reliance on WC field in packet header, interlaced support will not work if WC is selected from REGISTER mode.

29.16.141 CSI2_PIXEL_STREAM_A_GAP_0

CSI Pixel Stream A Gap

Offset: 0x611 | Byte Offset: 0x1844 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPA_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.

Bit	Reset	Description
15:0	0x0	PPA_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

29.16.142 CSI2_PIXEL_STREAM_PPA_COMMAND_0

CSI Pixel Parser A Command

Offset: 0x612 | Byte Offset: 0x1848 | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPA_START_MARKER_FRAME_MAX: CSI Pixel Parser A Start Marker Maximum. Start Frame is indicated when Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.
11:8	X	CSI_PPA_START_MARKER_FRAME_MIN: CSI Pixel Parser A Start Marker Minimum. Start Frame is indicated when Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.
4	X	CSI_PPA_VSYNC_START_MARKER: CSI Pixel Parser A 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPA_START_MARKER_FRAME_MIN and less than or equal to CSI_PPA_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC: VSYNC Start Marker start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC.
2	0x1	CSI_PPA_SINGLE_SHOT: CSI Pixel Parser A Single Shot Mode. Software should clear it along with disabling the CSI_PPA_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPA_ENABLE: CSI Pixel Parser A Enable. This parameter controls CSI Pixel Parser A to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST: reset (disable immediately). Enabling the pixel parser does not enable the corresponding input source to receive data. If pixel parser is enabled later than the corresponding input source, the CSI will keep on rejecting incoming streams, until it encounters a valid SF.

29.16.143 CSI2_PIXEL_STREAM_A_EXPECTED_FRAME_0

CSI Pixel Stream A Expected Frame

Offset: 0x613 | Byte Offset: 0x184c | Read/Write: R/W | Reset: 0x0000XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:4	X	PPA_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	0x0	PPA_ENABLE_LINE_TIMEOUT: When set to one enables checking of the time between start line requests from the Header Parser to CSI-PPA. A fake EF will be outputted by CSI-PPA if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines outputted, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPA_PAD_FRAME.

29.16.144 CSI2_CSI_PIXEL_PARSER_A_INTERRUPT_MASK_0

CSI Pixel Parser A Interrupt Mask

Offset: 0x614 | Byte Offset: 0x1850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPA_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPA_UNC_HDR_ERR. 0 = DISABLED: Don't generate an interrupt when HPA_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPA_UNC_HDR_ERR is set.
10	0x0	PPA_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPA_SPARE_STATUS_1. 0 = DISABLED: Don't generate an interrupt when PPA_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPA_SPARE_STATUS_1 is set.
9	0x0	PPA_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPA_INTERFRAME_LINE. 0 = DISABLED: Don't generate an interrupt when PPA_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPA_INTERFRAME_LINE is set.
8	0x0	PPA_EXTRA_SF_INT_MASK: Interrupt Mask for PPA_EXTRA_SF. 0 = DISABLED: Don't generate an interrupt when PPA_EXTRA_SF is set. 1 = ENABLED: Generate an interrupt when PPA_EXTRA_SF is set.
7	0x0	PPA_SHORT_FRAME_INT_MASK: Interrupt Mask for PPA_SHORT_FRAME. 0 = DISABLED: Don't generate an interrupt when PPA_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPA_SHORT_FRAME is set.
6	0x0	PPA_STMERR_INT_MASK: Interrupt Mask for PPA_STMERR. 0 = DISABLED: Don't generate an interrupt when PPA_STMERR is set. 1 = ENABLED: Generate an interrupt when PPA_STMERR is set.
5	0x0	PPA_FIFO_OVRF_INT_MASK: Interrupt Mask for PPA_FIFO_OVRF. 0 = DISABLED: Don't generate an interrupt when PPA_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPA_FIFO_OVRF is set.
4	0x0	PPA_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPA_PL_CRC_ERR. 0 = DISABLED: Don't generate an interrupt when PPA_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPA_PL_CRC_ERR is set.
3	0x0	PPA_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPA_SL_PKT_DROPPED. 0 = DISABLED: Don't generate an interrupt when PPA_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PKT_DROPPED is set.
2	0x0	PPA_SL_PROCESSED_INT_MASK: Interrupt Mask for PPA_SL_PROCESSED. 0 = DISABLED: Don't generate an interrupt when PPA_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPA_SL_PROCESSED is set.
1	0x0	PPA_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPA_ILL_WD_CNT. 0 = DISABLED: Don't generate an interrupt when PPA_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPA_ILL_WD_CNT is set.
0	0x0	PPA_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPA_HDR_ERR_COR. 0 = DISABLED: Don't generate an interrupt when PPA_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPA_HDR_ERR_COR is set.

29.16.145 CSI2_CSI_PIXEL_PARSER_A_STATUS_0

Pixel Parser A Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_PIXEL_PARSER_STATUS will clear only PPA_ILL_WD_CNT.

Offset: 0x615 | Byte Offset: 0x1854 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	HPA_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser A parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected. The packet will be discarded.
10	X	PPA_SPARE_STATUS_1: PPA Spare Status bit. This bit will get set when Pixel Parser A has a line timeout. Line timeout needs to be enabled by setting PPA_ENABLE_LINE_TIMEOUT and programming PPA_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPA_INTERFRAME_LINE: Set when CSI-PPA receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.

Bit	Reset	Description
8	X	PPA_EXTRA_SF: Set when CSI-PPA receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPA will insert a fake EF and the drop the current frame with Correct SF.
7	X	PPA_SHORT_FRAME: Set when CSI-PPA receives a short frame. This bit gets set even if CSI_PPA_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPA_STMERR: Stream Error. Set when the control output of PPA doesn't follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
5	X	PPA_FIFO_OVRF: FIFO Overflow. Set when the FIFO that is feeding packets to PPA overflows.
4	X	PPA_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPA had a payload CRC error.
3	X	PPA_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_A_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPA_SL_PROCESSED: Short Line Processed, Set when a line with a payload that is shorter than its packet header word count is processed by PPA.
1	X	PPA_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that doesn't generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPA.
0	X	PPA_HDR_ERR_COR: Header Error Corrected, Set when a packet that was processed by PPA has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_A_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

29.16.146 CSI2_CSI_SW_SENSOR_A_RESET_0

Offset: 0x616 | Byte Offset: 0x1858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_A_RESET: Reset CSI sensor A

29.16.147 CSI2_INPUT_STREAM_B_CONTROL_0

CSI Input Stream B Control

CSI PPB group

Offset: 0x61b | Byte Offset: 0x186c | Read/Write: R/W | Reset: 0x00ff0004 (0bxxxxxxx011111111xxxxxxxxxx0x100)

Bit	Reset	Description
24:16	0xff	CSI_B_SKIP_PACKET_THRESHOLD: CSI-B Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_B_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE
2	0x1	CSI_B_BYPASS_ALIGN: Bypass aligning CSIB lanes if valids for the lanes are out of sync by a cycle
1:0	0x0	CSI_B_DATA_LANE: CSI-B Data Lane 0= 1 data lane 1= 2 data lanes 2= 3 data lanes (not supported) 3= 4 data lanes (not supported) Note: 3 data lanes is not supported in Test Pattern Generation mode

29.16.148 CSI2_PIXEL_STREAM_B_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x61c | Byte Offset: 0x1870 | Read/Write: R/W | Reset: 0xXXXX01X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx000)

Bit	Reset	Description
29:28	X	CSI_PPB_PAD_FRAME: CSI Pixel Parser B Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPB_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPB_WORD_COUNT needs to be set to the number of input bytes in each lines payload. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD: Short frames will not be padded out.
27	X	CSI_PPB_HEADER_EC_ENABLE: CSI Pixel Parser B Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected, or not. 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE: Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser B. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB).
25:24	X	CSI_PPB_PAD_SHORT_LINE: CSI Pixel Parser B Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count) 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD: short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receives CSI output stream to hang up.
21:20	X	CSI_PPB_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser B Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPB_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by other CSI pixel stream processor. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED: output embedded data as 8-bpp arbitrary data stream.
19:16	X	CSI_PPB_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser B Output Format Options. This parameter specifies output data format. 0 = ARBITRARY: Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE: Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed.
8	0x1	CSI_PPB_WC_CHECK: CSI Pixel Parser B Data WC Check. This parameter specifies whether the word-count is checked. 0 = DISABLE: Word count Check is disabled 1 = ENABLE: Word count Check is enabled
7	X	CSI_PPB_CRC_CHECK: CSI Pixel Parser B Data CRC Check. This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE: Data CRC Check is enabled.
6	X	CSI_PPB_WORD_COUNT_SELECT: CSI Pixel Parser B Word Count Select. This parameter is effective only if packet header is sent as part of the stream. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPB_WORD_COUNT is different than the count in the packet header. Interlaced support relies on frame number and will not work in REGISTER mode. 1 = HEADER: The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. It is recommended to always program this bit to HEADER.

Bit	Reset	Description
5	X	CSI_PPB_DATA_IDENTIFIER: CSI Pixel Parser B Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPB_DATA_TYPE and against CSI_PPB_VIRTUAL_CHANNEL_ID). In this case, CSI_PPB_DATA_TYPE specifies the stream data format. 1 = ENABLED: Enabled - Data Identifier byte in packet header should be compared against the CSI_PPB_DATA_TYPE and the CSI_PPB_VIRTUAL_CHANNEL_ID.
4	X	CSI_PPB_PACKET_HEADER: CSI Pixel Parser B Packet Header processing. This specifies whether packet header is sent in the beginning of packet or not. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPB_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT: Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.
2:0	0x0	CSI_PPB_STREAM_SOURCE: CSI Pixel Parser B Stream Source 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B

29.16.149 CSI2_PIXEL_STREAM_B_CONTROL1_0

CSI Pixel Stream B Control 1

Offset: 0x61d | Byte Offset: 0x1874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	CSI_PPB_TOP_FIELD_FRAME_MASK: CSI Pixel Parser B Top Field Frame Mask
3:0	0x0	CSI_PPB_TOP_FIELD_FRAME: CSI Pixel Parser B Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise OR of ~(CSI_PPB_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPB_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet. Due to the reliance on WC field in packet header, interlaced support won't work if WC is selected from REGISTER mode.

29.16.150 CSI2_PIXEL_STREAM_B_GAP_0

CSI Pixel Stream B Gap

Offset: 0x61e | Byte Offset: 0x1878 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	PPB_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPB_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

29.16.151 CSI2_PIXEL_STREAM_PPB_COMMAND_0

CSI Pixel Parser B Command

Offset: 0x61f | Byte Offset: 0x187c | Read/Write: R/W | Reset: 0x0000XXX4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
15:12	X	CSI_PPB_START_MARKER_FRAME_MAX: CSI Pixel Parser B Start Marker Maximum. Start Frame is indicated when Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.

Bit	Reset	Description
11:8	X	CSI_PPB_START_MARKER_FRAME_MIN: CSI Pixel Parser B Start Marker Minimum. Start Frame is indicated when Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.
4	X	CSI_PPB_VSYNC_START_MARKER: CSI Pixel Parser B VSYNC Start Marker 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPB_START_MARKER_FRAME_MIN and less than or equal to CSI_PPB_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC: Start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC.
2	0x1	CSI_PPB_SINGLE_SHOT: CSI Pixel Parser B Single Shot Mode. Software should clear it along with disabling the CSI_PPB_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPB_ENABLE: CSI Pixel Parser B Enable. This parameter controls CSI Pixel Parser B to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST: reset (disable immediately). Enabling the pixel parser does not enable the corresponding input source to receive data. If Pixel parser is enabled later than the corresponding input source, the CSI will keep on rejecting incoming streams, until it encounters a valid SF.

29.16.152 CSI2_PIXEL_STREAM_B_EXPECTED_FRAME_0

CSI Pixel Stream B Expected Frame

Offset: 0x620 | Byte Offset: 0x1880 | Read/Write: R/W | Reset: 0x0000XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:4	X	PPB_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	0x0	PPB_ENABLE_LINE_TIMEOUT: When set to one enables checking of the time between start line requests from the Header Parser to CSI-PPB. A fake EF will be outputted by CSI-PPB if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines outputted, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPB_PAD_FRAME.

29.16.153 CSI2_CSI_PIXEL_PARSER_B_INTERRUPT_MASK_0

CSI Pixel Parser B Interrupt Mask

Offset: 0x621 | Byte Offset: 0x1884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0xx000000000000)

Bit	Reset	Description
14	0x0	HPB_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPB_UNC_HDR_ERR. 0 = DISABLED: Don't generate an interrupt when HPB_UNC_HDR_ERR is set. 1 = ENABLED: Generate an interrupt when HPB_UNC_HDR_ERR is set.
10	0x0	PPB_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPB_SPARE_STATUS_1. 0 = DISABLED: Don't generate an interrupt when PPB_SPARE_STATUS_1 is set. 1 = ENABLED: Generate an interrupt when PPB_SPARE_STATUS_1 is set.
9	0x0	PPB_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPB_INTERFRAME_LINE. 0 = DISABLED: Don't generate an interrupt when PPB_INTERFRAME_LINE is set. 1 = ENABLED: Generate an interrupt when PPB_INTERFRAME_LINE is set.
8	0x0	PPB_EXTRA_SF_INT_MASK: Interrupt Mask for PPB_EXTRA_SF. 0 = DISABLED: Don't generate an interrupt when PPB_EXTRA_SF is set. 1 = ENABLED: Generate an interrupt when PPB_EXTRA_SF is set.
7	0x0	PPB_SHORT_FRAME_INT_MASK: Interrupt Mask for PPB_SHORT_FRAME. 0 = DISABLED: Don't generate an interrupt when PPB_SHORT_FRAME is set. 1 = ENABLED: Generate an interrupt when PPB_SHORT_FRAME is set.
6	0x0	PPB_STMERR_INT_MASK: Interrupt Mask for PPB_STMERR. 0 = DISABLED: Don't generate an interrupt when PPB_STMERR is set. 1 = ENABLED: Generate an interrupt when PPB_STMERR is set.

Bit	Reset	Description
5	0x0	PPB_FIFO_OVRF_INT_MASK: Interrupt Mask for PPB_FIFO_OVRF. 0 = DISABLED: Don't generate an interrupt when PPB_FIFO_OVRF is set. 1 = ENABLED: Generate an interrupt when PPB_FIFO_OVRF is set.
4	0x0	PPB_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPB_PL_CRC_ERR. 0 = DISABLED: Don't generate an interrupt when PPB_PL_CRC_ERR is set. 1 = ENABLED: Generate an interrupt when PPB_PL_CRC_ERR is set.
3	0x0	PPB_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPB_SL_PKT_DROPPED. 0 = DISABLED: Don't generate an interrupt when PPB_SL_PKT_DROPPED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PKT_DROPPED is set.
2	0x0	PPB_SL_PROCESSED_INT_MASK: Interrupt Mask for PPB_SL_PROCESSED. 0 = DISABLED: Don't generate an interrupt when PPB_SL_PROCESSED is set. 1 = ENABLED: Generate an interrupt when PPB_SL_PROCESSED is set.
1	0x0	PPB_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPB_ILL_WD_CNT. 0 = DISABLED: Don't generate an interrupt when PPB_ILL_WD_CNT is set. 1 = ENABLED: Generate an interrupt when PPB_ILL_WD_CNT is set.
0	0x0	PPB_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPB_HDR_ERR_COR. 0 = DISABLED: Don't generate an interrupt when PPB_HDR_ERR_COR is set. 1 = ENABLED: Generate an interrupt when PPB_HDR_ERR_COR is set.

29.16.154 CSI2_CSI_PIXEL_PARSER_B_STATUS_0

Pixel Parser B Status

Offset: 0x622 | Byte Offset: 0x1888 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	HPB_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser B parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected.
10	X	PPB_SPARE_STATUS_1: PPB Spare Status bit. This bit will get set when Pixel Parser B has a line timeout. Line timeout needs to be enabled by setting PPB_ENABLE_LINE_TIMEOUT and programming PPB_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPB_INTERFRAME_LINE: Set when CSI-PPB receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.
8	X	PPB_EXTRA_SF: Set when CSI-PPB receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPB will insert a fake EF and the drop the current frame with Correct SF.
7	X	PPB_SHORT_FRAME: Set when CSI-PPB receives a short frame. This bit gets set even if CSI_PPB_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPB_STMERR: Stream Error. Set when the control output of PPB doesn't follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
5	X	PPB_FIFO_OVRF: FIFO Overflow. Set when the FIFO that is feeding packets to PPB overflows.
4	X	PPB_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPB had a payload CRC error.
3	X	PPB_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_B_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPB_SL_PROCESSED: Short Line Processed, Set when a line with a payload that is shorter than its packet header word count is processed by PPB.
1	X	PPB_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that doesn't generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPB.
0	X	PPB_HDR_ERR_COR: Header Error Corrected, set when a packet that was processed by PPB has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_B_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

29.16.155 CSI2_CSI_SW_SENSOR_B_RESET_0

Offset: 0x623 | Byte Offset: 0x188c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_SENSOR_B_RESET: Reset CSI sensor B

29.16.156 CSI2_PHY_CIL_COMMAND_0

CSI Phy and CIL Command

CSI CIL group

Offset: 0x642 | Byte Offset: 0x1908 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx00xxxxxx00)

Bit	Reset	Description
9:8	0x0	CSI_B_PHY_CIL_ENABLE: CSI B Phy and CIL Enable. This parameter controls CSI B Phy and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: disable (reset)
1:0	0x0	CSI_A_PHY_CIL_ENABLE: CSI A Phy and CIL Enable. This parameter controls CSI A Phy and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: disable (reset)

29.16.157 CSI2_CIL_PAD_CONFIG0_0

CIL Pad Configuration 0

Offset: 0x643 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000xxxxxxxx)

Bit	Reset	Description
15:8	0x0	PAD_CIL_SPARE: Spare bit for CIL BIAS Config. PAD_CIL_SPARE[7] is used is being used to flush VI's Y-FIFO when it is being use as a stream source for one of the Pixel Parsers. Setting PAD_CIL_SPARE[7] to 1 will hold vi2csi_host_stall low. Which will force VI's Y-FIFO to be purged. PAD_CIL_SPARE[7] must be low for the pixel parser to receive source data from VI's Y-FIFO.

29.16.158 CSI2_CILA_PAD_CONFIG0_0

CIL-A Pad Configuration 0

CSI CILA group

Offset: 0x64b | Byte Offset: 0x192c | Read/Write: R/W | Reset: 0x00010007 (0b0000x000000000010000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILA_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. will cause stress and need waiver to use
30:28	0x0	PAD_CILA_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
26:24	0x0	PAD_CILA_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
23:21	0x0	PAD_CILA_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only
20:18	0x0	PAD_CILA_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only

Bit	Reset	Description
17:16	0x1	PAD_AB_BK_MODE: 00: two independent 2x bricks 01: one 4x brick, received clock from partition A is used. Clock from partition B is not used 10: one 4x brick, received clock from partition B is used. Clock from partition A is not used 11: illegal
15	0x0	PAD_CILA_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILA_INADJ1: bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILA_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILA_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILA_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
2	0x1	PAD_CILA_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILA_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

29.16.159 CSI2_CILA_PAD_CONFIG1_0

CIL-A Pad Configuration 4

Offset: 0x64c | Byte Offset: 0x1930 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILA_SPARE_RO: Spare Read only bits for CILA Config
15:8	RW	0x0	PAD_CILA_SPARE: Spare bits for CILA Config. PAD_CILA_SPARE[15] is being used to disable the CSI-A RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled, 1: push blocking enabled
7:6	RW	0x0	PAD_CILA_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
5:4	RW	0x0	PAD_CILA_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
3:2	RW	0x0	PAD_CILA_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
1:0	RW	0x0	PAD_CILA_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only

29.16.160 CSI2_PHY_CILA_CONTROL0_0

CSI-A Phy and CIL Control

Offset: 0x64d | Byte Offset: 0x1934 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILA_CLK_SETTLE: settle time for clk lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz lp clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 csicil cycles for default value. So the programming value (0 to 15) of this field works like this 0 = 15 cilclk cycles (default internal delay), 1 = 15 - 7 (8 cilclk cycles), 2 = 15 - 6 (9 cilclk cycles), . . 7 = 15 - 1 (14 cilclk cycles), 8 = 15 - 0 (default internal delay), 9 = 15 + 1 (16 cilclk cycles), . 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILA_BYPASS_LP_SEQ: The LP signals should sequence through LP11->LP01->LP00 state, to indicate to CLOCK CIL about the mode switching to HS Rx mode. In case the camera is enabled earlier than CIL, it is highly likely that camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
5:0	0x2	CILA_THS_SETTLE: settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

29.16.161 CSI2_CSI_CIL_A_INTERRUPT_MASK_0

CSI Control and Interface Logic A Interrupt Mask

Offset: 0x64e | Byte Offset: 0x1938 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
9	0x0	CILA_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CLK_CTRL_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_CLK_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CLK_CTRL_ERR is set.
6	0x0	CILA_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILA_ESC_DATA_REC. 0 = DISABLED: Don't generate an interrupt when CILA_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_DATA_REC is set.
5	0x0	CILA_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILA_ESC_CMD_REC. 0 = DISABLED: Don't generate an interrupt when CILA_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILA_ESC_CMD_REC is set.
4	0x0	CILA_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CTRL_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_CTRL_ERR is set.
2	0x0	CILA_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILA_SYNC_ESC_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SYNC_ESC_ERR is set.
1	0x0	CILA_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_MB_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_MB_ERR is set.
0	0x0	CILA_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_SB_ERR. 0 = DISABLED: Don't generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILA_SOT_SB_ERR is set.

29.16.162 CSI2_CSI_CIL_A_STATUS_0

CSI Control and Interface Logic A Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_A_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 0x64f | Byte Offset: 0x193c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILA_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-A receives an Escape Mode Data byte. The Data Byte can be read from bits 7-0 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILA_ESC_CMD_REC is set.
5	X	CILA_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-A receives an Escape Mode Command byte. The Command Byte can be read from bits 7-0 of ESCAPE_MODE_COMMAND.
4	X	CILA_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
2	X	CILA_SYNC_ESC_ERR: Sync Escape Error. Set when CIL-A detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
1	X	CILA_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packets SOT bytes. The packet will be discarded.
0	X	CILA_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

29.16.163 CSI2_CSI_CILA_STATUS_0

CSI-CILA Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x650 | Byte Offset: 0x1940 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILA_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILA_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-1. The packet will be discarded.
16	X	CILA_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-A for processing.
6	X	CILA_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILA_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-0. The packet will be discarded.
4	X	CILA_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-A for processing.
0	X	CILA_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-A detects incorrect line state sequence on clk lane

29.16.164 CSI2_CIL_A_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-A.

Offset: 0x651 | Byte Offset: 0x1944 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILA_ESC_CMD_BYTE: CIL-A Escape Mode Command Byte. This is the 8 bit entry command that was received, by CIL-A, during the last escape Mode sequence. CIL-A monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILA_ESC_CMD_REC status bit is set.

29.16.165 CSI2_CIL_A_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-A.

Offset: 0x652 | Byte Offset: 0x1948 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILA_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte. When read this field returns the last Escape Mode Data byte that was received by CIL-A. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILA_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte comes in.

29.16.166 CSI2_CSICIL_SW_SENSOR_A_RESET_0

Offset: 0x653 | Byte Offset: 0x194c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_A_RESET: Reset CSICIL sensor A

29.16.167 CSI2_CILB_PAD_CONFIG0_0

CIL-B Pad Configuration 0

CSI CILB group

Offset: 0x658 | Byte Offset: 0x1960 | Read/Write: R/W | Reset: 0x00000007 (0b0000x000000000xx0000x000x0000111)

Bit	Reset	Description
31	0x0	PAD_CILB_E_TXBW: 0 = default to enable VCLAMP regulator 1 = turn off regulator and short vclamp to VDDP. will cause stress and need waiver to use
30:28	0x0	PAD_CILB_SLEWDNADJ: Pull down slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
26:24	0x0	PAD_CILB_SLEWUPADJ: Pull up slew rate adjust, default 000. From 000 -> 011, slew rate increases. 100 is the same as 000. From 100->111, skew rate decreases. This control is unused for CSI MIPI pads, which are receive only
23:21	0x0	PAD_CILB_LPDNADJ: Driver pull down impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only
20:18	0x0	PAD_CILB_LPUPADJ: Driver pull up impedance control. 00 -> 130ohm, default. 01 -> 110ohm. 10 -> 130ohm, same as 00. 11 -> 150ohm. This control is unused for CSI MIPI pads, which are receive only
15	0x0	PAD_CILB_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILB_INADJ1: bit 1 input delay trimmer, each tap delays 20ps
10:8	0x0	PAD_CILB_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILB_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILB_PDVCLAMP: Power down regulator which supplies current to serializer/de-serializer logic
2	0x1	PAD_CILB_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILB_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

29.16.168 CSI2_CILB_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 0x659 | Byte Offset: 0x1964 | Read/Write: R/W | Reset: 0xXXXX0000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILB_SPARE_RO: Spare Read only bits for CILB Config
15:8	RW	0x0	PAD_CILB_SPARE: Spare bits for CILB Config. PAD_CILB_SPARE[15] is being used to disable the CSI-B RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled, 1: push blocking enabled
7:6	RW	0x0	PAD_CILB_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
5:4	RW	0x0	PAD_CILB_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
3:2	RW	0x0	PAD_CILB_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only
1:0	RW	0x0	PAD_CILB_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00=no pre-emphasis, 11=max. This control is unused for CSI MIPI pads, which are receive only

29.16.169 CSI2_PHY_CILB_CONTROL0_0

CSI-B Phy and CIL Control

Offset: 0x65a | Byte Offset: 0x1968 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxx000000x0000010)

Bit	Reset	Description
13:8	0x0	CILB_CLK_SETTLE: settle time for clk lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 csicil cycles for default value. So the programming value (0 to 15) of this field works like this 0 = 15 cilclk cycles (default internal delay), 1 = 15 - 7 (8 cilclk cycles), 2 = 15 - 6 (9 cilclk cycles), . . 7 = 15 - 1 (14 cilclk cycles), 8 = 15 - 0 (default internal delay), 9 = 15 + 1 (16 cilclk cycles), . 57 = 15 + 49 (64 cilclk cycles)
6	0x0	CILB_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
5:0	0x2	CILB_THS_SETTLE: settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

29.16.170 CSI2_CSI_CIL_B_INTERRUPT_MASK_0

CSI Control and Interface Logic B Interrupt Mask

Offset: 0x65b | Byte Offset: 0x196c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0x0000x000)

Bit	Reset	Description
6	0x0	CILB_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILB_ESC_DATA_REC. 0 = DISABLED: Don't generate an interrupt when CILB_ESC_DATA_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_DATA_REC is set.
5	0x0	CILB_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILB_ESC_CMD_REC. 0 = DISABLED: Don't generate an interrupt when CILB_ESC_CMD_REC is set. 1 = ENABLED: Generate an interrupt when CILB_ESC_CMD_REC is set.
4	0x0	CILB_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CTRL_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_CTRL_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_CTRL_ERR is set.
2	0x0	CILB_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILB_SYNC_ESC_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_SYNC_ESC_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SYNC_ESC_ERR is set.
1	0x0	CILB_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_MB_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_SOT_MB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_MB_ERR is set.
0	0x0	CILB_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_SB_ERR. 0 = DISABLED: Don't generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED: Generate an interrupt when CILB_SOT_SB_ERR is set.

29.16.171 CSI2_CSI_CIL_B_STATUS_0

CSI Control and Interface Logic B Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_B_STATUS will clear only CILB_SOT_MB_ERR.

Offset: 0x65c | Byte Offset: 0x1970 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	CILB_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-B receives an Escape Mode Data byte. The Data Byte can be read from bits 23-16 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILB_ESC_CMD_REC is set.
5	X	CILB_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-B receives an Escape Mode Command byte. The Command Byte can be read from bits 23-16 of ESCAPE_MODE_COMMAND.
4	X	CILB_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).

Bit	Reset	Description
2	X	CILB_SYNC_ESC_ERR: Sync Escape Error. Set when CIL-B detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
1	X	CILB_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packets SOT bytes. The packet will be discarded.
0	X	CILB_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-B detects a single bit error in one of the packets start of transmission bytes. The packet will be sent to CSI-B for processing.

29.16.172 CSI2_CSI_CILB_STATUS_0

CSI-CILB Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x65d | Byte Offset: 0x1974 | Read/Write: RO | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	CILB_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILB_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-1. The packet will be discarded.
16	X	CILB_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-B for processing.
6	X	CILB_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILB_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi bit start of transmission byte error in one of the packets SOT bytes on data lane-0. The packet will be discarded.
4	X	CILB_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-B detects a single bit error in one of the packets Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-B for processing.
0	X	CILB_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-B detects incorrect line state sequence on clk lane

29.16.173 CSI2_CIL_B_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-B.

Offset: 0x65e | Byte Offset: 0x1978 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILB_ESC_CMD_BYTE: CIL-B Escape Mode Command Byte. This is the 8 bit entry command that was received, by CIL-B, during the last escape Mode sequence. CIL-B monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILB_ESC_CMD_REC status bit is set.

29.16.174 CSI2_CIL_B_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-B.

Offset: 0x65f | Byte Offset: 0x197c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CILB_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte. When read, this field returns the last Escape Mode Data byte that was received by CIL-B. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILB_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte comes in.

29.16.175 CSI2_CSICIL_SW_SENSOR_B_RESET_0

Offset: 0x660 | Byte Offset: 0x1980 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSICIL_SENSOR_B_RESET: Reset CSICIL sensor B

29.16.176 CSI2_PATTERN_GENERATOR_CTRL_A_0

CSI common group

Offset: 0x671 | Byte Offset: 0x19c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_A: Mode for Sensor A 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_A: Automatic phase increment mode for sensor A
0	0x0	PG_ENABLE_A: Enable Pattern Generator for sensor A

29.16.177 CSI2_PG_BLANK_A_0

Offset: 0x672 | Byte Offset: 0x19c8 | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_A: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_A: Horizontal Blanking for PG

29.16.178 CSI2_PG_PHASE_A_0

Offset: 0x673 | Byte Offset: 0x19cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_A: Initial Phase

29.16.179 CSI2_PG_RED_FREQ_A_0

Offset: 0x674 | Byte Offset: 0x19d0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.180 CSI2_PG_RED_FREQ_RATE_A_0

Offset: 0x675 | Byte Offset: 0x19d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_A: Rate of change of vertical frequency

Bit	Reset	Description
7:0	0x0	PG_RED_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.181 CSI2_PG_GREEN_FREQ_A_0

Offset: 0x676 | Byte Offset: 0x19d8 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.182 CSI2_PG_GREEN_FREQ_RATE_A_0

Offset: 0x677 | Byte Offset: 0x19dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.183 CSI2_PG_BLUE_FREQ_A_0

Offset: 0x678 | Byte Offset: 0x19e0 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_A: Initial horizontal frequency

29.16.184 CSI2_PG_BLUE_FREQ_RATE_A_0

Offset: 0x679 | Byte Offset: 0x19e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

29.16.185 CSI2_PG_AOHDR_A_0

Offset: 0x67a | Byte Offset: 0x19e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:1	0x0	PG_AOHDR_GAIN_RATIO_A: Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ration (short rows >> 2)
0	0x0	PG_AOHDR_ENABLE_A: AOHDR enable

29.16.186 CSI2_PATTERN_GENERATOR_CTRL_B_0

Offset: 0x67e | Byte Offset: 0x19f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:2	0x0	PG_MODE_B: Mode for Sensor B 0 = DIRECT 1 = PATCH 2 = RSVD
1	0x0	PG_AUTO_INC_B: Automatic phase increment mode for sensor B
0	0x0	PG_ENABLE_B: Enable Pattern Generator for sensor B

29.16.187 CSI2_PG_BLANK_B_0

Offset: 0x67f | Byte Offset: 0x19fc | Read/Write: R/W | Reset: 0x00080008 (0b00000000000010000000000000001000)

Bit	Reset	Description
31:16	0x8	PG_VBLANK_B: Vertical Blanking for PG
15:0	0x8	PG_HBLANK_B: Horizontal Blanking for PG

29.16.188 CSI2_PG_PHASE_B_0

Offset: 0x680 | Byte Offset: 0x1a00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	PG_PHASE_B: Initial Phase

29.16.189 CSI2_PG_RED_FREQ_B_0

Offset: 0x681 | Byte Offset: 0x1a04 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.190 CSI2_PG_RED_FREQ_RATE_B_0

Offset: 0x682 | Byte Offset: 0x1a08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.191 CSI2_PG_GREEN_FREQ_B_0

Offset: 0x683 | Byte Offset: 0x1a0c | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.192 CSI2_PG_GREEN_FREQ_RATE_B_0

Offset: 0x684 | Byte Offset: 0x1a10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.193 CSI2_PG_BLUE_FREQ_B_0

Offset: 0x685 | Byte Offset: 0x1a14 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000xx00000000000000)

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_B: Initial horizontal frequency

29.16.194 CSI2_PG_BLUE_FREQ_RATE_B_0

Offset: 0x686 | Byte Offset: 0x1a18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

29.16.195 CSI2_PG_AOHDR_B_0

Offset: 0x687 | Byte Offset: 0x1a1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:1	0x0	PG_AOHDR_GAIN_RATIO_B: Gain ratio for channel balancing; Short exposure gain: Long Exposure gain 0: 2:1 Gain ratio (short rows <<1) 1: 4:1 Gain Ratio (short rows << 2) 2: 0.5:1 Gain Ratio (short rows >> 1) 3: 0.25:1 Gain Ration (short rows >> 2)
0	0x0	PG_AOHDR_ENABLE_B: AOHDR enable

29.16.196 CSI2_DPCM_CTRL_A_0

Offset: 0x68b | Byte Offset: 0x1a2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_A: DPCM A compression ratio 0: BYPASS 1: 10_8_10 2: 10_7_10 3: 10_6_10 4: 12_8_12 5: 12_7_12 6: 12_6_12 7: 14_10_14 8: 14_8_14
0	0x0	DPCM_PREDICTOR_A: DPCM A predictor 0: predictor1, 1: predictor2

29.16.197 CSI2_DPCM_CTRL_B_0

Offset: 0x68c | Byte Offset: 0x1a30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxx0)

Bit	Reset	Description
11:8	0x0	DPCM_COMPRESSION_RATIO_B: DPCM A compression ratio 0: BYPASS 1: 10_8_10 2: 10_7_10 3: 10_6_10 4: 12_8_12 5: 12_7_12 6: 12_6_12 7: 14_10_14 8: 14_8_14
0	0x0	DPCM_PREDICTOR_B: DPCM A predictor 0: predictor1, 1: predictor2

29.16.198 CSI2_STALL_COUNTER_0

Offset: 0x691 | Byte Offset: 0x1a44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	STALL_SENSOR_B_COUNT
7:0	0x0	STALL_SENSOR_A_COUNT: Number of cycles to stall sensor A after every EOF

29.16.199 CSI2_CSI_READONLY_STATUS_0

CSI Read Only Status

This register is used to return CSI read only status.

Offset: 0x692 | Byte Offset: 0x1a48 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	CSI_PPB_ACTIVE: One only when Pixel Parser B is capturing frame data.
0	X	CSI_PPA_ACTIVE: One only when Pixel Parser A is capturing frame data.

29.16.200 CSI2_CSI_SW_STATUS_RESET_0

Offset: 0x693 | Byte Offset: 0x1a4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CSI_STATUS_RESET: Reset CSI status and dbgcnt registers

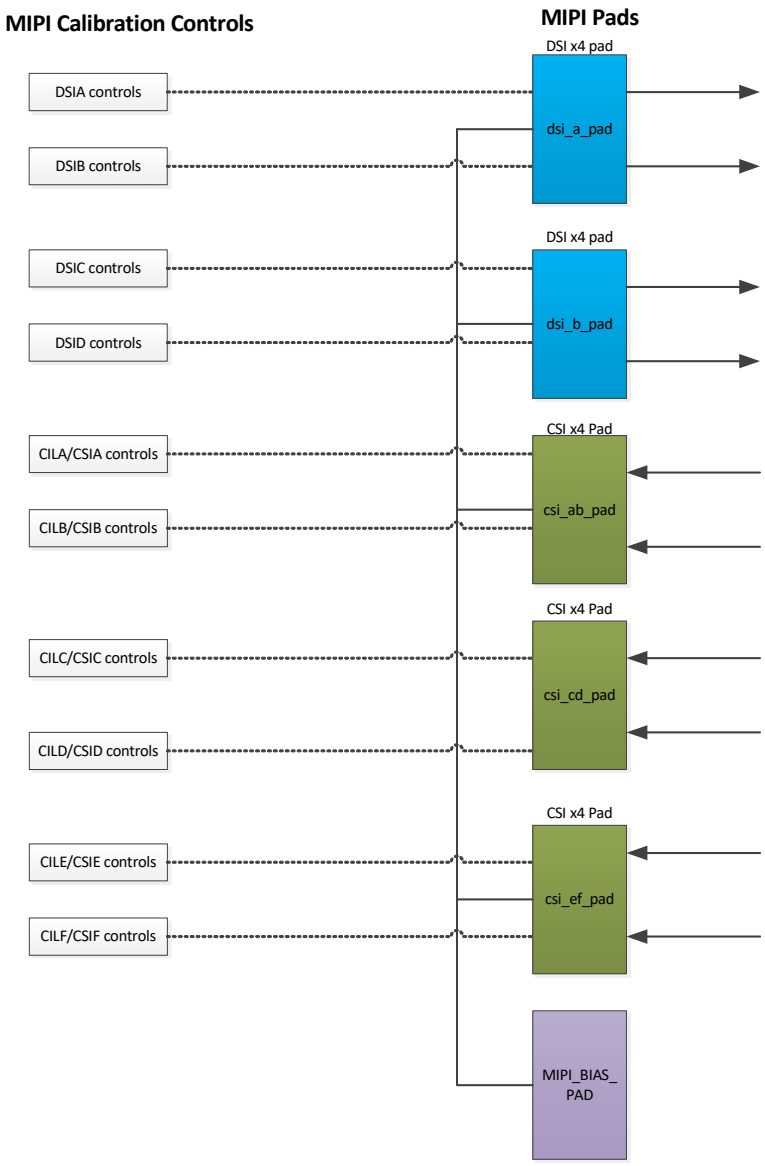
CHAPTER 30: MIPI D-PHY CALIBRATION FOR CSI AND DSI

This chapter describes the MIPI D-PHY Calibration registers. This information is provided to aid in system debug and diagnostics. It is expected that NVIDIA® supplied drivers will normally be used to program these.

MIPI D-PHY blocks are used for both the DSI display interface and the CSI camera interface. The calibration for both interfaces is controlled by these registers.

The MIPI pad calibration block controls a voltage regulator that must be powered up whenever any of the DSI pads are used. The regulator is controlled by the PAD_PDVREG bit in the MIPI_CAL_MIPI_BIAS_PAD_CFG2_0 register. Clearing this bit powers up the regulator while setting the bit powers it down.

Figure 124: Mapping of MIPI Calibration Control Registers to MIPI Pads



The figure above shows the channel assignments of the MIPI D-PHYs in Tegra® X1.

30.1 Implementation of DSI MIPI Calibration

30.1.1 Pre MIPI_CAL

1. Set MIPI_CAL_MIPI_BIAS_PAD_CFG0_0 and MIPI_CAL_MIPI_BIAS_PAD_CFG2_0

   MIPI_CAL_MIPI_BIAS_PAD_CFG0_0[0]: MIPI_BIAS_PAD_E_VCLAMP_REF = 1

   MIPI_CAL_MIPI_BIAS_PAD_CFG2_0 = 0
2. Reset MIPI_CAL and enable MIPI_CAL Clock

   ///SW reset MIPI_CAL

   REG(CLK_RST_CONTROLLER_RST_DEVICES_H_0,SWR_MIPI_CAL_RST,1)

   waitmsec(100)

   REG(CLK_RST_CONTROLLER_RST_DEVICES_H_0,SWR_MIPI_CAL_RST,0)

   //enable MIPI_CAL clock

   REG(CLK_RST_CONTROLLER_PLLP_OUTB_0, PLLP_OUT3_RATIO, 4)

   //If PLLP=216 MHz, PLLP_OUT3=72 MHz. If PLLP=408 MHz, set Ratio to 10 to have

   PLLP_OUT3=68 MHz

   REG(CLK_RST_CONTROLLER_PLLP_OUTB_0, PLLP_OUT3_CLKEN, 1)

   waitmsec(1)

   REG(CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0, CLK_ENB_MIPI_CAL, 1) //MIPI_CLK

   source PLLP_OUT3

30.1.2 MIPI_CAL Settings

MIPI_CAL shall be done with DSI_A, DSI_B and DSI_A + DSI_B (dual channel) based on the applications.

1. Use of DSI_A
   a. Select the lanes to be used

      reg(MIPI_CAL_DSIA_MIPI_CAL_CONFIG_0,MIPI_CAL_SELDSIA, 1) // first 2 lanes in DSI_A

      REG(MIPI_CAL_DSIB_MIPI_CAL_CONFIG_0,MIPI_CAL_SELDSIB,1) // 2nd 2 lanes in DSI_A

      REG(MIPI_CAL_DSIA_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELDSIA,1) //SEL_DSIA_CLK

      REG(MIPI_CAL_DSIB_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELDSIB,1) //SEL_DSIB_CLK

   b. De-select non-used MIPI lanes

      reg(MIPI_CAL_CILA_MIPI_CAL_CONFIG_0,MIPI_CAL_SELA, 0) //CILA

      reg(MIPI_CAL_CILB_MIPI_CAL_CONFIG_0,MIPI_CAL_SELB, 0) //CILB

      reg(MIPI_CAL_CILC_MIPI_CAL_CONFIG_0,MIPI_CAL_SELC, 0) //CILC

      reg(MIPI_CAL_CILD_MIPI_CAL_CONFIG_0,MIPI_CAL_SELD, 0) //CILD

      reg(MIPI_CAL_CILE_MIPI_CAL_CONFIG_0,MIPI_CAL_SELE, 0) //CILE

      reg(MIPI_CAL_CILC_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELC, 0) //CLKSELC

      reg(MIPI_CAL_CILD_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELD, 0) //CLKSELD

      reg(MIPI_CAL_CSIE_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELE, 0) //CLKSELE

   c. Set MIPI_CAL parameters

      MIPI_CAL_MIPI_BIAS_PAD_CFG1_0[10:8] = 3 // PAD_DRIV_UP_REF

      MIPI_CAL_MIPI_BIAS_PAD_CFG1_0[18:16] = 0 // PAD_DRIV_DN_REF

MIPI_CAL_DSIA_MIPI_CAL_CONFIG_2_0[4:0] = 2 // MIPI_CAL_HSCLKPUOSDSIA

MIPI_CAL_DSIA_MIPI_CAL_CONFIG_2_0[12:8] = 0 // MIPI_CAL_HSCLKPDOSDSIA

MIPI_CAL_DSIB_MIPI_CAL_CONFIG_2_0[4:0] = 2 // MIPI_CAL_HSCLKPUOSDSIB

MIPI_CAL_DSIB_MIPI_CAL_CONFIG_2_0[12:8] = 0 // MIPI_CAL_HSCLKPDOSDSIB

Other non-mentioned parameters keep with the default values.

d. DSI settings

DSI_PAD_CONTROL_3_0[1:0] = 3 //DSI_PAD_PREEMP_PU

DSI_PAD_CONTROL_3_0[5:4] = 3 //DSI_PAD_PREEMP_PD

DSI_PAD_CONTROL_3_0[9:8] = 3 //DSI_PAD_PREEMP_PU_CLK

e. Adjustment based on PCB trace and layout

Since individual customer PCB boards have different trace length and characterizations, it may need to adjust the clock and data delays in individual lanes with the following registers.

DSI_PAD_CONTROL_2_0[2:0] =>DSI_PAD_OUTADJCLK

DSI_PAD_CONTROL_1_0[2:0] => DSI_PAD_OUTADJ0

DSI_PAD_CONTROL_1_0[6:4] => DSI_PAD_OUTADJ1

DSI_PAD_CONTROL_1_0[10:8] => DSI_PAD_OUTADJ2

DSI_PAD_CONTROL_1_0[14:12] => DSI_PAD_OUTADJ3

Since it is PCB specific, it does not provide a definite value for the setting for the delay here. If there is noise with unclear border lines in the display image, it requires this adjustment.

2. Use of DSI_B

a. Switch the bus from CSI to DSI

REG(APB_MISC_GP_MIPI_PAD_CTRL_0, DSIB_MODE, 1) //0=CSI, 1 =DSIB

b. Reset DSIB

REG(CLK_RST_CONTROLLER_RST_DEVICES_U_0, SWR_DSIB_RST, 1)

waitmsec(20)

REG(CLK_RST_CONTROLLER_RST_DEVICES_U_0, SWR_DSIB_RST, 0)

REG(CLK_RST_CONTROLLER_RST_DEV_U_SET_0, SET_DSIB_RST, 1)

waitmsec(20)

REG(CLK_RST_CONTROLLER_RST_DEV_U_CLR_0, CLR_DSIB_RST, 1)

REG(CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0, CLK_ENB_DSIB, 1)

REG(CLK_RST_CONTROLLER_CLK_ENB_U_SET_0, SET_CLK_ENB_DSIB, 1)

c. Select the lanes to be used

reg(MIPI_CAL_CILC_MIPI_CAL_CONFIG_0,MIPI_CAL_SELC, 1) //CILC

reg(MIPI_CAL_CILD_MIPI_CAL_CONFIG_0,MIPI_CAL_SELD, 1) //CILD

reg(MIPI_CAL_CILC_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELC, 1) //CLKSELC

reg(MIPI_CAL_CILD_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELD, 1) //CLKSELD

d. De-select non-used MIPI lanes

reg(MIPI_CAL_CILA_MIPI_CAL_CONFIG_0,MIPI_CAL_SELA, 0) //CILA

reg(MIPI_CAL_CILB_MIPI_CAL_CONFIG_0,MIPI_CAL_SELB, 0) //CILB

reg(MIPI_CAL_CILE_MIPI_CAL_CONFIG_0,MIPI_CAL_SELE, 0) //CILE

reg(MIPI_CAL_CSIE_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELE, 0) //CLKSELE

reg(MIPI_CAL_DSIA_MIPI_CAL_CONFIG_0,MIPI_CAL_SELDSIA, 0)

REG(MIPI_CAL_DSIA_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELDSIA,0)

REG(MIPI_CAL_DSIB_MIPI_CAL_CONFIG_0,MIPI_CAL_SELDSIB,0)

REG(MIPI_CAL_DSIB_MIPI_CAL_CONFIG_2_0,MIPI_CAL_CLKSELDSIB,0)

e. Set MIPI_CAL parameters

MIPI_CAL_MIPI_BIAS_PAD_CFG1_0[10:8] = 3 // PAD_DRIV_UP_REF

MIPI_CAL_MIPI_BIAS_PAD_CFG1_0[18:16] = 0 // PAD_DRIV_DN_REF

MIPI_CAL_CILC_MIPI_CAL_CONFIG_2_0 [4:0] = 2 // MIPI_CAL_HSCLKPUOSCILC

MIPI_CAL_CILC_MIPI_CAL_CONFIG_2_0 [12:8] = 3 // MIPI_CAL_HSCLKPDOSCILC

MIPI_CAL_CILD_MIPI_CAL_CONFIG_2_0 [4:0] = 2 // MIPI_CAL_HSCLKPUOSCILD

MIPI_CAL_CILD_MIPI_CAL_CONFIG_2_0 [12:8] = 3 // MIPI_CAL_HSCLKPDOSCILD

Other non-mentioned parameters keep with the default values.

f. DSI settings

DSI_PAD_CONTROL_3_1[1:0] = 3 //DSI_PAD_PREEMP_PU

DSI_PAD_CONTROL_3_1[5:4] = 3 //DSI_PAD_PREEMP_PD

DSI_PAD_CONTROL_3_1[9:8] = 3 //DSI_PAD_PREEMP_PU_CLK

g. Adjustment based on PCB trace and layout

Since individual customer PCB boards have different trace length and characterizations, it may need to adjust the clock and data delays in individual lanes with the following registers.

DSI_PAD_CONTROL_2_1[2:0] =>DSI_PAD_OUTADJCLK

DSI_PAD_CONTROL_1_1[2:0] => DSI_PAD_OUTADJ0

DSI_PAD_CONTROL_1_1[6:4] => DSI_PAD_OUTADJ1

DSI_PAD_CONTROL_1_1[10:8] => DSI_PAD_OUTADJ2

DSI_PAD_CONTROL_1_1[14:12] => DSI_PAD_OUTADJ3

Since it is PCB specific, it does not provide a definite value for the setting for the delay here. If there is noise with unclear border lines in the display image, it requires this adjustment.

3. Use of DSI_A + DSI_B for dual channel

Do both steps for Use of DSI_A and DSI_B listed above in addition to ganged mode settings.

30.1.3 MIPI_CAL Enable

1. Enable MIPI_CAL

reg 0x700e3008 0xffffffff //clear status

REG(MIPI_CAL_MIPI_CAL_CTRL_0,MIPI_CAL_NOISE_FLT,0xa) //Filter

REG(MIPI_CAL_MIPI_CAL_CTRL_0,MIPI_CAL_PRESCALE,0x2) //pre-scale

REG(MIPI_CAL_MIPI_CAL_CTRL_0,MIPI_CAL_CLKEN_OVR,0x1) //clk_always_on

reg(MIPI_CAL_MIPI_CAL_CTRL_0,MIPI_CAL_AUTOCAL_EN, %autocal_en) ///if need keeping

MIPI_CAL, %autocal_en = 1 and %startcal = 0

reg(MIPI_CAL_MIPI_CAL_AUTOCAL_CTRL0_0,MIPI_CAL_AUTOCAL_PERIOD, 0xFFFFFFF) //

reg(MIPI_CAL_MIPI_CAL_CTRL_0,MIPI_CAL_STARTCAL, %startcal) //start autocal

2. Status Verification

   After MIPI_CAL enabled, it may need to check the status of MIPI_CAL by calling

   reg list MIPI_CAL_CIL_MIPI_CAL_STATUS_0* *

   reg list MIPI_CAL_CIL_MIPI_CAL_STATUS_2_0* *

30.2 MIPI-CAL Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

30.2.1 MIPI_CAL_MIPI_CAL_CTRL_0

Calibration Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x2a000000 (0bxx101010xxxxxxxxxxxxxxxxxxxx0xx00)

Bit	R/W	Reset	Description
29:26	RW	0xa	MIPI_CAL_NOISE_FLT: The DRIVRY and TERMRY signals coming from MIPI pads are utilized by the Calibration state machine for pad calibration. The drivry/termry comes from a noisy analog source, which might have some glitches. The filter in the Calibration state machine is sensitive to these noises. If the calibration done status does not show up, the sensitivity of the filter can be changed through these bits. Ideally, this has to be programmed in a range from 10 to 15. When MIPI_CAL_PRESCALE = 2'b00, this needs to be programmed between 2 to 5.
25:24	RW	0x2	MIPI_CAL_PRESCALE: Auto-Cal calibration step prescale: 00: when calibration step is 0.1 µs 01: when calibration step is 0.5 µs 10: when calibration step is 1.0 µs (default) 11: when calibration step is 1.5 µs This keeps the MIPI bias calibration step between 0.1-1.5 µs.
4	RW	CLK_GATED	MIPI_CAL_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
1	RW	0x0	MIPI_CAL_AUTOCAL_EN: When set, calibration is triggered periodically. The Period is set using MIPI_CAL_AUTOCAL_CTRL0 register
0	RO	0x0	MIPI_CAL_STARTCAL: Writing a one to this bit starts the Calibration State machine. This bit must be set even if the TERM/HSPU/HSPD values need to be overridden using the TERMOS/HSPUOS/HSPDOS register fields.

30.2.2 MIPI_CAL_MIPI_CAL_AUTOCAL_CTRL0_0

Offset: 0x4 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:0	-1	MIPI_CAL_AUTOCAL_PERIOD: Auto-calibration period in number of APB system clock cycles.

30.2.3 MIPI_CAL_CIL_MIPI_CAL_STATUS_0

CIL MIPI Calibration Status

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	MIPI_AUTO_CAL_DONE_DSID: Debug bit. MIPI Auto Calibration done for DSID, set when the auto calibration sequence for DSID pad brick is done. Write 1-to-clear.
30	X	MIPI_AUTO_CAL_DONE_DSIC: Debug bit. MIPI Auto Calibration done for DSIC, set when the auto calibration sequence for DSIC pad brick is done. Write 1-to-clear.
29	X	MIPI_AUTO_CAL_DONE_DSIB: Debug bit. MIPI Auto Calibration done for DSIB. Set when the auto calibration sequence for DSIA (upper two data lanes) pad brick is done. Write 1-to-clear.

Bit	Reset	Description
28	X	MIPI_AUTO_CAL_DONE_DSIA: Debug bit. MIPI Auto Calibration done for DSI. Set when the auto calibration sequence for DSIA (upper two data lanes) pad brick is done. Write 1-to-clear.
25	X	MIPI_AUTO_CAL_DONE_CSIF: Debug bit. MIPI Auto Calibration done for CSI, set when the auto calibration sequence for CSIF pad brick is done. Write 1-to-clear.
24	X	MIPI_AUTO_CAL_DONE_CSIE: Debug bit. MIPI Auto Calibration done for CSI, set when the auto calibration sequence for CSIE pad brick is done. Write 1-to-clear.
23	X	MIPI_AUTO_CAL_DONE_CSID: Debug bit. MIPI Auto Calibration done for CSI. Set when the auto calibration sequence for shared CSID/DSIB (upper two data lanes) pad brick is done. Write 1-to-clear.
22	X	MIPI_AUTO_CAL_DONE_CSIC: Debug bit. MIPI Auto Calibration done for CSI. Set when the auto calibration sequence for shared CSIC/DSIB (lower two data lanes) pad brick is done. Write 1-to-clear.
21	X	MIPI_AUTO_CAL_DONE_CSIB: Debug bit. MIPI Auto Calibration done for CSI. Set when the auto calibration sequence for CSIB pad brick is done. Write 1-to-clear.
20	X	MIPI_AUTO_CAL_DONE_CSIA: Debug bit. MIPI Auto Calibration done for CSI. Set when the auto calibration sequence for CSIA pad brick is done. Write 1-to-clear.
16	X	MIPI_AUTO_CAL_DONE: Debug bit. MIPI Auto Calibration done. Set when the auto-calibration sequence for all the selected CSI/DSI MIPI pad bricks is done. Write 1-to-clear.
15:12	X	MIPI_CAL_DRIV_DN_ADJ: Debug bit. Driver Pull-down calibration code generated by MIPI auto calibration. Valid only after the auto-calibration sequence has completed (MIPI_CAL_ACTIVE == 0).
11:8	X	MIPI_CAL_DRIV_UP_ADJ: Debug bit. Driver Pull up calibration code generated by MIPI auto calibration. Valid only after auto calibration sequence has completed (MIPI_CAL_ACTIVE == 0).
7:4	X	MIPI_CAL_TERMADJ: Debug bit. Termination code generated by MIPI auto calibration. Valid only after the auto-calibration sequence has completed (MIPI_CAL_ACTIVE == 0).
0	X	MIPI_CAL_ACTIVE: Debug bit. One when auto calibration is active.

30.2.4 MIPI_CAL_CIL_MIPI_CAL_STATUS_2_0

MIPI CLK Calibration Status 2

Offset: 0xc | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	MIPI_CLK_AUTO_CAL_DONE_DSID: MIPI CLK Auto Calibration done for DSID, set when the auto calibration sequence for DSIB pad's CLK (upper) lane is done. Write 1-to-clear.
5	X	MIPI_CLK_AUTO_CAL_DONE_DSIC: MIPI CLK Auto Calibration done for DSIC, set when the auto calibration sequence for DSIB pad's CLK (lower) lane is done. Write 1-to-clear.
4	X	MIPI_CLK_AUTO_CAL_DONE_DSIB: MIPI CLK Auto Calibration done for DSIB, set when the auto calibration sequence for DSIA pad's CLK (upper) lane is done. Write 1-to-clear.
3	X	MIPI_CLK_AUTO_CAL_DONE_DSIA: MIPI CLK Auto Calibration done for DSIA, set when the auto calibration sequence for DSIA pad's CLK (lower) lane is done. Write 1-to-clear.

30.2.5 MIPI_CAL_CILA_MIPI_CAL_CONFIG_0

Calibration Settings for CSI-ABMIPI Pad’s Channel A (lower two data lanes)

Offset: 0x14 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx1xxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEA: When 0 (normal operation), use the value in the TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel A TERMADJ/HSPUADJ/ HSPDADJ values to the MIPI pads. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to channel A TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELA: Select the CSI-AB Pad's channel A data lanes for auto calibration.
4:0	0x0	MIPI_CAL_TERMOSA: 2's complement offset for TERMADJ going to channel A (currently negative offsets are not supported)

30.2.6 MIPI_CAL_CILB_MIPI_CAL_CONFIG_0

Calibration Settings for CSI-ABMIPI Pad’s Channel B (upper two data lanes)

Offset: 0x18 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEB: When 0 (normal operation), use the value in the TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel B TERMADJ/HSPUADJ/ HSPDADJ values to the MIPI pads. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to channel B TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELB: Select the CSI-AB Pad's channel A data lanes for auto calibration.
4:0	0x0	MIPI_CAL_TERMOSB: 2's complement offset for TERMADJ going to channel B (currently negative offsets are not supported)

30.2.7 MIPI_CAL_CILC_MIPI_CAL_CONFIG_0

Calibration Settings for CSI-ABMIPI Pad’s Channel B (upper two data lanes)

Offset: 0x1c | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEC: When 0 (normal operation), use the value in the TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel C TERMADJ/HSPUADJ/ HSPDADJ values to the MIPI pads. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to channel C TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELC: Select the CSI-AB Pad's channel B data lanes for auto calibration.
4:0	0x0	MIPI_CAL_TERMOSC: 2's complement offset for TERMADJ going to channel C (currently negative offsets are not supported)

30.2.8 MIPI_CAL_CILD_MIPI_CAL_CONFIG_0

Calibration Settings for Shared CSI-CD MIPI Pad’s channel B (upper two data lanes)

Offset: 0x20 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDED: When 0 (normal operation), use the value in the TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel B TERMADJ/HSPUADJ/ HSPDADJ values to the MIPI pads. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to channel B TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELD: Select the CSI-CD Pad's channel B data lanes for auto calibration.
4:0	0x0	MIPI_CAL_TERMOSD: 2's complement offset for TERMADJ going to channel B (currently negative offsets are not supported)

30.2.9 MIPI_CAL_CILE_MIPI_CAL_CONFIG_0

Calibration Settings for CSI-EF MIPI Pad’s Channel A (lower two data lanes

Offset: 0x24 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEE: When 0 (normal operation), use the value in the TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel A TERMADJ/HSPUADJ/ HSPDADJ values to the MIPI pads. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to channel A TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELE: Select the CSIEF pad’s channel A data lanes for auto calibration.
4:0	0x0	MIPI_CAL_TERMOSE: 2's complement offset for TERMADJ going to channel E (currently negative offsets are not supported)

30.2.10 MIPI_CAL_CILF_MIPI_CAL_CONFIG_0

Calibration Settings for CSI-EF MIPI pad's channel B (upper two data lanes)

Offset: 0x28 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEF: When 0 (normal operation), use the value in TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel E TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pad. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to channel E TERMADJ/HSPUADJ/HSPDADJ on the MIPI pad.
21	0x1	MIPI_CAL_SELF: Select the CSIEF pad's channel B data lanes for auto calibration.
4:0	0x0	MIPI_CAL_TERMOSF: 2's complement offset for TERMADJ going to channel B (Currently negative offsets are not supported)

30.2.11 MIPI_CAL_DSIA_MIPI_CAL_CONFIG_0

Calibration Settings for DSIA MIPI Pad's Channel A (lower two data lanes)

Offset: 0x38 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIA: When 0 (normal operation), use the value in the TERMOS/HSPUOS/ HSPDOS register fields as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/ HSPDADJ values to the MIPI pads. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELDSIA: Select the DSIA pad's Channel A data lanes for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIA: 2's complement offset for HSPDADJ going to channel A (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSDSIA: 2's complement offset for HSPUADJ going to channel A (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSDSIA: 2's complement offset for TERMADJ going to channel A (currently negative offsets are not supported)

30.2.12 MIPI_CAL_DSIB_MIPI_CAL_CONFIG_0

Calibration Settings for DSIA MIPI Pad’s Channel B (upper two data lanes)

Offset: 0x3c | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIB: When 0 (normal operation), use the value in the TERMOS/HSPUOS/ HSPDOS register fields as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/ HSPDADJ values to the MIPI pads. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELDSIB: Select the DSIA pad's Channel B data lanes for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIB: 2's complement offset for HSPDADJ going to channel B (currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSDSIB: 2's complement offset for HSPUADJ going to channel B (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSDSIB: 2's complement offset for TERMADJ going to channel B (currently negative offsets are not supported)

30.2.13 MIPI_CAL_DSIC_MIPI_CAL_CONFIG_0

Calibration settings for DSIB MIPI pad's Channel A (lower two data lanes)

Offset: 0x40 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIC: When 0 (normal operation), use the value in TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pad. When 1, use the register values in TERMOS/HSPUOS/HSPDOS as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pad.
21	0x1	MIPI_CAL_SELDSIC: Select the DSIB pad's Channel A data lanes for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIC: 2's complement offset for HSPDADJ going to channel A (Currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSDSIC: 2's complement offset for HSPUADJ going to channel A (Currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSDSIC: 2's complement offset for TERMADJ going to channel A (Currently negative offsets are not supported)

30.2.14 MIPI_CAL_DSID_MIPI_CAL_CONFIG_0

Calibration settings for DSIB MIPI pad's Channel B (upper two data lanes)

Offset: 0x44 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSID: When 0 (normal operation), use the value in TERMOS/HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pad. When 1, use the value in TERMOS/HSPUOS/HSPDOS as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pad.
21	0x1	MIPI_CAL_SELDSID: Select the DSIB pad's Channel B data lanes for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSID: 2's complement offset for HSPDADJ going to channel B (Currently negative offsets are not supported)
12:8	0x0	MIPI_CAL_HSPUOSDSID: 2's complement offset for HSPUADJ going to channel B (Currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_TERMOSDSID: 2's complement offset for TERMADJ going to channel B (Currently negative offsets are not supported)

30.2.15 MIPI_CAL_MIPI_BIAS_PAD_CFG0_0

MIPI Bias Pad Configuration Register

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MIPI_BIAS_PAD_PDVCLAMP: 1=Power down regulator which supplies current to pre-driver logic.
0	0x0	MIPI_BIAS_PAD_E_VCLAMP_REF: 0=The reference voltage internal regulator is merged with Bandgap voltage generator (preferred setting) 1=Generates reference voltage for internal regulator from resistor divider.

30.2.16 MIPI_CAL_MIPI_BIAS_PAD_CFG1_0

MIPI Bias Pad Configuration Register

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx000xxxxx000xxxxx000xxxxx000)

Bit	Reset	Description
26:24	0x0	PAD_TEST_SEL: Controls which signal to be routed to TEST_OUT 000=VAUXP 001=RUP 010=RDN 011=VREG 1xx=TBD
18:16	0x0	PAD_DRIV_DN_REF: Adjust internal reference level for drive pull-down calibration
10:8	0x0	PAD_DRIV_UP_REF: Adjust internal reference level for drive pull-up calibration
2:0	0x0	PAD_TERM_REF: Adjust internal reference level for termination calibration

30.2.17 MIPI_CAL_MIPI_BIAS_PAD_CFG2_0

MIPI Bias Pad Configuration Register 2

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxx00000000000x000x010)

Bit	Reset	Description
18:16	0x0	PAD_VCLAMP_LEVEL: VCLAMP level adjustment
15:8	0x0	PAD_SPARE: Spare bit for MIPI Bias Config
6:4	0x0	PAD_VAUXP_LEVEL: VAUXP level adjustment 00 = no adjustment, default 01 = 105% 10 = 110% 11 = 115% 100 = no adjustment 101 = 95% 110 = 90% 111 = 85%
2	0x0	PAD_E_TXBW: This bit should not be used. Set it to 0.
1	0x1	PAD_PDVREG: Power down the MIPI DSI and CSI pad supply voltage regulator. 0: power up supply voltage regulator 1: power down supply voltage regulator
0	0x0	PAD_VBYPASS: Bypass bang gap voltage reference

30.2.18 MIPI_CAL_DSIA_MIPI_CAL_CONFIG_2_0

Calibration Settings for DSIA pad Channel A CLK (lower CLK lane)

Offset: 0x64 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEDSIA: When 0 (normal operation), use the value in HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel A HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values in HSPUOS/HSPDOS as the actual values going to channel A HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELDSIA: Select the DSIA PAD Channel A clock lane for auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSDSIA: 2's complement offset for HSCLKPDADJ going to Channel A CLK lane (currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_HSCLKPUOSDSIA: 2's complement offset for HSCLKPUADJ going to Channel A CLK lane (Currently negative offsets are not supported)

30.2.19 MIPI_CAL_DSIB_MIPI_CAL_CONFIG_2_0

Calibration Settings for DSIA pad's Channel B CLK (upper CLK lane)

Offset: 0x68 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEDSIB: When 0 (normal operation), use the value in HSPUOS/HSPDOS register fields as an offset to the Calibration State machine setting for channel B HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values in HSPUOS/HSPDOS as the actual values going to channel B HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELDSIB: Select the DSIA PAD Channel B clock lane for auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSDSIB: 2's complement offset for HSCLKPDADJ going to Channel B CLK lane (Currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_HSCLKPUOSDSIB: 2's complement offset for HSCLKPUADJ going to Channel B CLK lane (Currently negative offsets are not supported)

30.2.20 MIPI_CAL_DSIC_MIPI_CAL_CONFIG_2_0

Calibration Settings for DSIB pad's Channel A CLK (lower CLK lane)

Offset: 0x70 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEDSIC: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for channel A HSCLKPUADJ/HSCLKPDADJ values to the MIPI Pads. When 1, use the register values above as the actual values going to channel A HSCLKPUADJ/ HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELDSIC: Select the DSIB PAD Channel A clock lane for auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSDSIC: 2's complement offset for HSCLKPDADJ going to Channel A CLK lane (Currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_HSCLKPUOSDSIC: 2's complement offset for HSCLKPUADJ going to Channel A CLK lane (Currently negative offsets are not supported)

30.2.21 MIPI_CAL_DSID_MIPI_CAL_CONFIG_2_0

Calibration Settings for DSIB pad's Channel B CLK (upper CLK lane)

Offset: 0x74 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx1xxxxxxxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_CLKOVERIDEDSID: When 0 (normal operation), use the value in the HSPUOS/HSPDOS register field as an offset to the Calibration State machine setting for channel B HSCLKPUADJ/ HSCLKPDADJ values to the MIPI Pads. When 1, use the register values above as the actual values going to channel B HSCLKPUADJ/HSCLKPDADJ on the MIPI Pads.
21	0x1	MIPI_CAL_CLKSELDSID: Select the DSIB PAD Channel B clock lane for auto calibration.
12:8	0x0	MIPI_CAL_HSCLKPDOSDSID: 2's complement offset for HSCLKPDADJ going to Channel B CLK lane (Currently negative offsets are not supported)
4:0	0x0	MIPI_CAL_HSCLKPUOSDSID: 2's complement offset for HSCLKPUADJ going to Channel B CLK lane (Currently negative offsets are not supported)

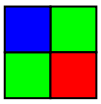
CHAPTER 31: VIDEO INPUT (VI)

The Video Input unit (VI3) receives data from the CSI unit and prepares it for presentation to system memory or the dedicated ISP execution resources. The Video Input unit provides formatting for RGB, YCbCr, and raw Bayer data in support of a number of camera user models. These models include single and multi-camera systems, which may have up to six active streams. The input streams are obtained from MIPI compliant CMOS sensor camera modules.

In addition to packing of image data, several advanced use cases are supported with processing of the raw Bayer data. These user modes involve the capture of high-resolution still images while simultaneously maintaining a lower resolution preview or video capture through the ISP2 in addition to the capture of stereo video and still images. In order to provide for improved synchronization of camera sensor control and the rest of the camera capture logic, a second physical interface to the Host1x is provided along with a dedicated I^2C interface as a major feature upgrade for the VI3 unit.

This chapter also describes the VI registers; however, it is not intended to be a programming guide to VI, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

31.1 VI Glossary

Term	Definition
Bayer	A type of image sampling pattern invented by Dr. Bryce E. Bayer of Eastman Kodak. The pattern consists of quads of pixels with two green samples, one red sample and one blue sample: 
EOF	End of Frame
RGB	Name given to pixels with Red Green and Blue color components. This is the pixel format typically found in most display technologies since each color component corresponds to the colors of the filters or phosphors used in the display device.
SOF	Start of Frame
YCbCr	An alternative pixel representation that takes advantage of the properties of the human psycho-perceptual vision system. It consists of a luminance channel Y and two color difference signals Cb and Cr.
YUV	See YCbCr. U and V are equivalent to Cb and Cr, respectively.
ZSL	Zero Shutter Lag

31.2 Functionality

31.2.1 Camera Subsystem

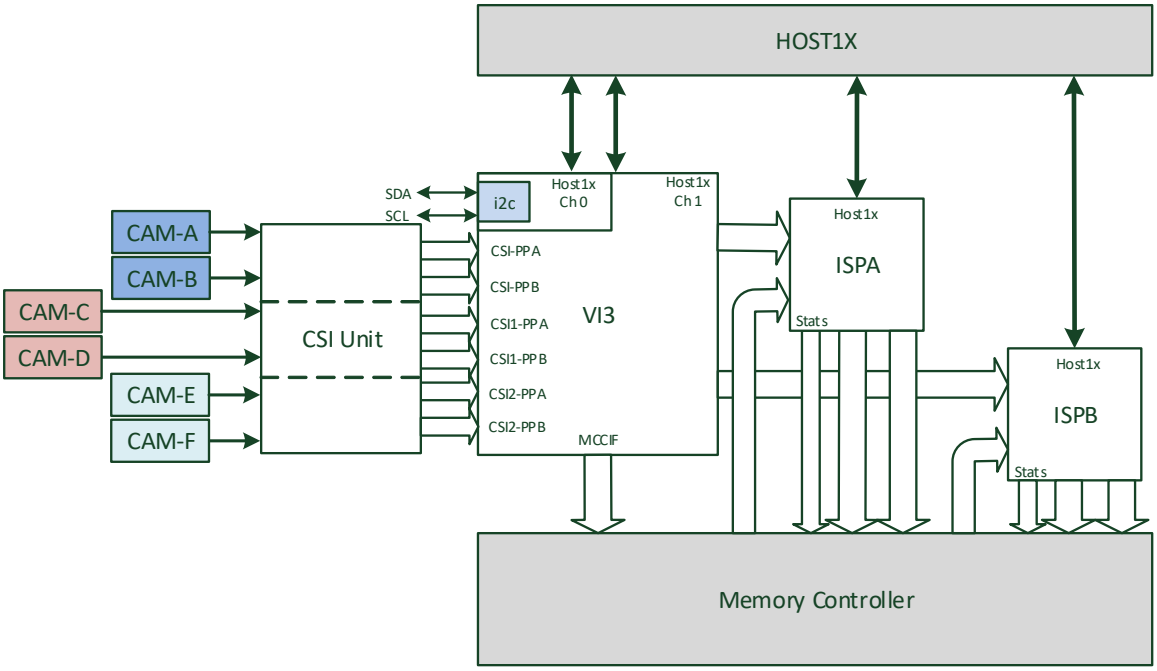
The VI3 unit interfaces with four major functional units within the Tegra® X1 camera subsystem in addition to the Host1x, which is used for control programming:

- Camera Serial Interface (CSI) – Provides for the interface to up to six MIPI compliant CMOS sensors through the CSI, CSI1 and CSI2 instances and formats the pixel data for presentation to the VI3 unit.
- Image Signal Processors A and B (ISPA and ISPB) – Execution resource providing image processing in support of various camera use cases and raw Bayer processing.
- Memory Controller (MC) – Interface between the VI3 unit and ISP and memory resources.

The main paths for the flow of image data for the VI3 unit are from the CSI → VI3 → ISPA/ISPB and from the CSI → VI3 → Memory Controller. There are also several additional paths for data flow within the camera's subsystem, which includes paths from the VI3 unit to Memory Controller and the Memory Controller to the ISP. The ability to present data to system memory

allows customer-specific algorithms, such as blur reduction, chromatic aberration correction, and high ISO noise processing, to be inserted into the ISP3.1 pipeline. An added benefit is the ability to divide input data into more manageable widths when processing high-resolution still images. The high-level block diagram for the Tegra X1 camera subsystem is illustrated in the following figure.

Figure 125: Tegra X1 Camera Subsystem



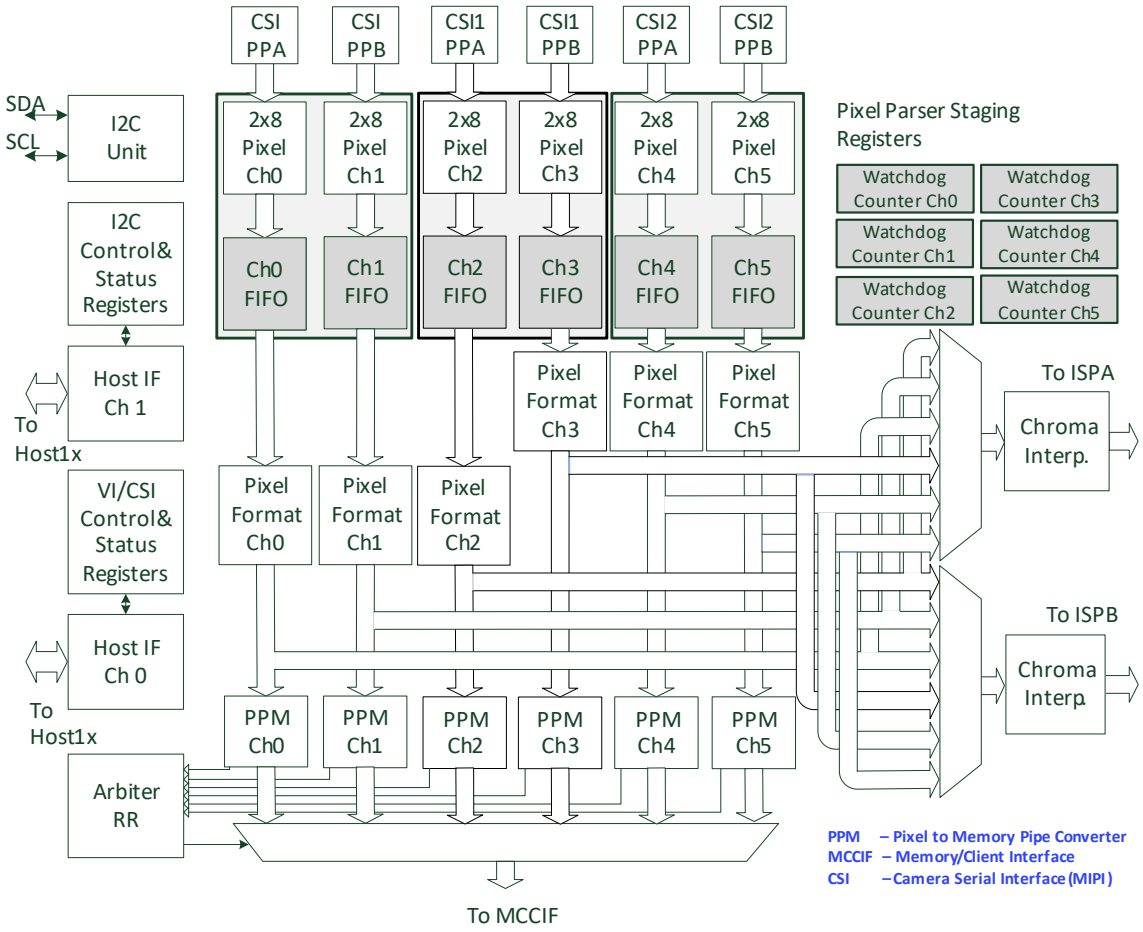
The support for multiple paths to and from memory within the camera subsystem is useful for very wide surfaces, such as still capture, where the alternative would be to have very large on-chip line stores. Because there are two ISP resources, two data streams from separate sensors or camera modules can be provided for immediate processing in support of dual and stereo camera capture models. In addition to the basic multi-camera user modes, the output path from the VI3 unit to memory allows the maintenance of a Zero Shutter Lag (ZSL) circular buffer for high-resolution capture while the ISP is processing the preview stream for display.

Two Host1x interfaces are provided within the VI3 unit. The first Host1x interface handles the flow of the programmed control state for the VI3 unit. The second Host1x interface provides a dedicated low latency control for camera support hardware through the I2C. The I2C module is embedded within the VI3 unit and may be programmed with either the Host1x or through Memory Mapped I/O. The CSI registers are physically located in the VI3 unit and are accessed through the Host interface of the VI3 unit. The I2C registers and interrupts are accessed through the dedicated Host1x. Refer to Chapter 29: MIPI-CSI (Camera Serial Interface) in this document for more information.

31.2.2 Block Diagram

The VI3 unit accepts pixel data from an external source, packages the received data, and presents the data to either system memory or the ISP resources directly. The pixel data received can be in a variety of formats including Raw, RGB, and YCbCr. The high-level block diagram for the VI3 unit is shown in the following figure.

Figure 126: VI3 High-Level Block Diagram



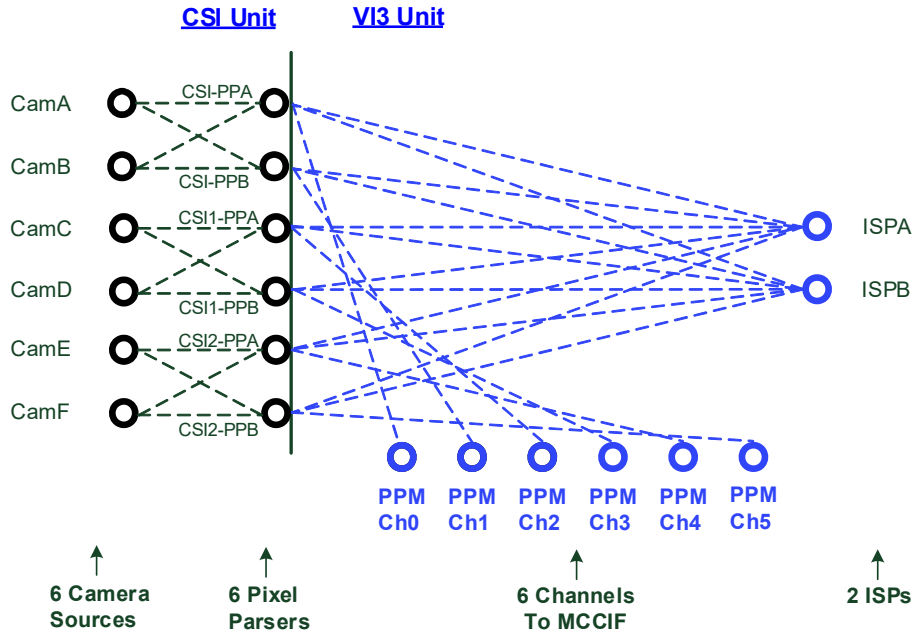
The VI3 unit offers these features not present in earlier Tegra devices:

- Four additional channels to allow up to six simultaneous streams to be managed
- An I^2C interface with a dedicated Host1x channel for control

31.2.3 Single and Multiple Stream Datapaths

The Tegra X1 camera subsystem provides considerable flexibility in the support of various single and dual camera user models as in previous implementations. In addition, to the single and dual camera support an expanded capability is introduced to manage multiple camera models with support for data capture from up to six camera sources. In all of the use cases, either one or two of the streams may be presented directly to the ISP resources in addition to system memory. As a result of having two ISP execution resources in the system, any capture scenario involving more than two camera sources will require management of at least one through presentation to system memory. The support for up to six camera streams introduces a flexibility that is manifested as a number of programmable configurations that allow the image data to be directed to system memory and ISP execution resources. The datapaths for the camera subsystem are illustrated in the following figure.

Figure 127: Multi-Stream Datapaths through the CSI and VI3



The CSI unit provides for connection of up to six cameras in the system and is organized as three identical instances of two MIPI support blocks, each with a separate 4-lane interface that can be configured as a single camera with 4 lanes or as a dual camera with 2 lanes available for each camera. Each of the two camera interfaces has two pixel parsers which provide the data feed to the VI3 unit. The VI3 unit can be programmed to route the data streams to system memory, the ISP resources, or both the ISP and system memory using the various configurations options available. In addition to supporting a dual stream capture from two sources, the CSI unit supports a broadcast mode in which a single camera data stream can be broadcast to both pixel parsers simultaneously for each of the instances (CSI-PPA and CSI-PPB, CSI1-PPA and CSI1-PPB, and CSI2-PPA and CSI2-PPB). In the broadcast mode the same data stream can be presented to the ISP or memory resources in a number of different ways.

There are many configurations possible for the single, dual, and three to six camera user models with the primary constraint that at most only two streams may be presented to the ISPA and/or ISPB with all other streams being routed to system memory. As with the previous VI implementations, a support for broadcast to both ISPs and/or system memory is supported. The following table provides the configuration options for a single camera source.

Table 176: Datapath Options through VI3 for Single Camera User Models

Datapath Active for Single Camera Source	Comments
ISPA	Single stream capture through either CSI-PPA, CSI-PPB, CSI1-PPA, CSI1-PPB, CSI2-PPA, or CSI2-PPB
ISPB	Single stream capture through either CSI-PPA, CSI-PPB, CSI1-PPA, CSI1-PPB, CSI2-PPA, or CSI2-PPB
ISPA & ISPB	Single stream broadcast through either (CSI-PPA and CSI-PPB), or (CSI1-PPA and CSI1-PPB), or (CSI2-PPA and CSI2-PPB),
ISPA & ISPB & [(PPM0 or PPM1)or (PPM2 or PPM3) or (PPM4 or PPM5)]	Single stream broadcast through either (CSI-PPA and CSI-PPB), or (CSI1-PPA and CS1I-PPB), or (CSI2-PPA and CSI2-PPB) ; single stream to MEM through either (PPM0 or PPM1) or (PPM2 or PPM3) or (PPM4 or PPM5)].
ISPA & ISPB & [(PPM0 & PPM1)or (PPM2 & PPM3) or (PPM4 & PPM5)]	Single stream broadcast through either (CSI-PPA and CSI-PPB), or (CSI1-PPA and CSI1-PPB), or (CSI2-PPA and CSI2-PPB); both ISPA and ISPB and either (PPM0 and PPM1) or (PPM2 and PPM3) or (PPM4 and PPM5) active.
ISPA & [(PPM0 or PPM1)or (PPM2 or PPM3) or (PPM4 or PPM5)]	Single stream capture through either (CSI-PPA or CSI-PPB), or (CSI1-PPA or CSI1-PPB), or (CSI2-PPA or CSI2-PPB); single stream to ISPA and MEM through either (PPM0 or PPM1) or (PPM2 or PPM3) or (PPM4 or PPM5).
ISPA & [(PPM0 & PPM1)or (PPM2 & PPM3) or (PPM4 & PPM5)]	Single stream capture through both either (CSI-PPA and CSI-PPB), or (CSI1-PPA and CSI1-PPB), or (CSI2-PPA and CSI2-PPB), single stream to ISPA and MEM through either (PPM0 and PPM1) or (PPM2 and PPM3) or (PPM4 and PPM5)

Table 176: Datapath Options through VI3 for Single Camera User Models

Datapath Active for Single Camera Source	Comments
ISPB & [(PPM0 or PPM1)or (PPM2 or PPM3) or (PPM4 or PPM5)]	Single stream capture through either (CSI-PPA and CSI-PPB), or (CSI1-PPA and CSI1-PPB), or (CSI2-PPA and CSI2-PPB), single stream to ISPB and MEM through either (PPM0 and PPM1) or (PPM2 and PPM3) or (PPM4 and PPM5)
(PPM0 or PPM1)or (PPM2 or PPM3) or (PPM4 or PPM5)	Single stream capture through either (CSI-PPA or CSI-PPB), or (CSI1-PPA or CSI1-PPB), or (CSI2-PPA or CSI2-PPB); single stream to MEM through either (PPM0 or PPM1) or (PPM2 or PPM3) or (PPM4 or PPM5).
(PPM0 & PPM1)or (PPM2 & PPM3) or (PPM4 & PPM5)	Single stream capture through either (CSI-PPA or CSI-PPB), or (CSI1-PPA or CSI1-PPB), or (CSI2-PPA or CSI2-PPB); single stream to MEM through either (PPM0 and PPM1) or (PPM2 and PPM3) or (PPM4 and PPM5)

The multiple camera use cases follow a similar set of configuration options as seen with the single camera. The system limitation of two ISP resources and the separate instances for the CSI unit introduce some constraints. The constraints are primarily that only two streams can be presented directly to the ISPA and ISPB simultaneously, and the broadcast capability is constrained to the grouping associated with CSI-PPA and CSI-PPB, CSI1-PPA and CSI1-PPB, and CSI2-PPA and CSI2-PPB.

31.2.4 Host1x Interfaces

The Host1x interface is the primary method for communicating control state from the CPU to the VI3 unit. It is responsible for the register write and read requests. Two Host1x interfaces are available. The first is used to configure the registers for to manage the camera data streaming from the CSI, CSI1, and CSI2 instances within the CSI unit to memory or the ISPA/B, and the second Host1x interface is provided to manage the I^2C transactions provided with the VI_I2C block.

The Tegra X1 camera subsystem targets single and multiple image sources for still and video capture as in previous Tegra devices. The VI3 unit has increased the number of camera streams which can be managed simultaneously to a maximum of six and added the VI_I2C as a means of issuing I^2C transactions. As a result of the increase in camera streams and the inclusion of the I^2C Controller within the VI3 unit, the number of host channels has also been increased to four.

31.2.4.1 Primary Host1x Interface

The Host1x interface of the VI3 unit is used to configure the registers for the VI3 unit and the CSI, CSI1, and CSI2 instances within the CSI unit. The rest of the section also describes some of the CSI related logic.

The registers can be categorized in two categories: configuration registers and operational registers. The configuration registers should remain constant for the life of the camera operation (until the camera unit is re-enabled between application switching). The operational registers are required to process a given frame of image.

The registers can be divided into two categories: configuration registers and operational registers. The configuration registers remain constant for the life of the camera operation (until the camera unit is re-enabled between application switching). The operational registers are required to process a given frame of image.

In order to process the images with minimum V-blank, the operational registers need to be double-buffered to speed up the transition between the operational parameters between the two frames.

VI State Updates

The operation registers are initially assembled in the assembly state. The registers are copied to the active state when the pipe is idle or immediately after startup. In all other cases, they are copied to the active state when the update command is received and when the next EOF is received. The following table describes the VI state update trigger.

Table 177: Single Shot Register Definition

Register	Field	Bits	Reset	Description
VI_CSI_[0..5]_SINGLE_SHOT	CAPTURE	0	0	Software sets this bit to request transition of the previously assembled assembly state to the active state and schedule single shot capture. The actual update of the operational register is gated by the next EOF. Register read back returns the current capture status 0: No capture pending. 1: Capture pending
VI_CSI_[0..5]_SINGLE_SHOT_UPDATE	CAPTURE_GOOD_FRAME	0	1	This bit controls when the update command should be processed. 1'b0: Transition to update when the next EOF is received, irrespective of the state of previous frame. **Note1:** Useful for debugging and/or bring up when the sensor/VI configuration could be incorrect. 1'b1: Transition to next operational state when the next EOF is received AND the previous frame was the correct frame based on the current operational state. **Note2:** Any mismatch will result in the frame getting dropped and the CSI rearming with unchanged configuration. **Note3**: After exiting the reset state, the previous frame with the correct state should be set.

In the normal functional mode of operation, the EOF updates the pending assembly state to the active state only if the previous frame was successfully captured (or this is the first frame out of reset). If for any reason the previous frame was detected to be an errored frame, on the next SOF, the VI3 and CSI units continue to work on the previous operational state configuration and do not update to the next set of operational parameters. An additional configuration register controls the behavior of VI3 unit.

VI Sync Points

Sync points (syncpts) are a means of synchronizing software programmed control state and hardware operations, without the need for register polling or interrupt service routines. Host1x contains a number of counters. These counters are referenced through an ***index***. Software can issue two kinds of syncpt-related methods:

- INCR_SYNCPT ***<condition> <index>***
- WAIT_SYNCPT ***<index> <threshold>***

INCR_SYNCPT tells the hardware that when ***<condition>*** is met, the hardware should send the value of ***<index>*** to the Host1x on the RtnIndx bus. On receiving the index return, Host1x will increment the counter pointed to by ***<index>***.

WAIT_SYNCPT is a Host1x method which tells Host1x to cease processing commands and register writes for that unit until the value of the counter pointed to by ***<index>*** reaches or exceeds the value of ***<threshold>***.

Since software can be made to wait for the counter to reach a certain value and hardware controls the incrementing of the counter, the two can be kept synchronized, ensuring that control state is metered out at the appropriate times by Host1x. In order to support the single shot and shadow register behaviors, sync point conditions 0x03 through 0x1F are buffered, allowing up to two sync points for each of these conditions to be programmed. The following table describes the VI3 sync point conditions.

Table 178: Sync Point Conditions

Condition	Value	FIFO Depth	Description
IMMEDIATE	0x00	1	Predefined Immediate condition. The VI3 returns this syncpt immediately.
OP_DONE	0x01	1	Predefined Operation Done Condition
RD_DONE	0x02	1	Predefined Read Done condition. The VI3 treats this as IMMEDIATE because CSI/VI do not have any read ports.
REG_WR_SAFE	0x03	2	Predefined Register Write Safe condition. Technically software should know when it is safe to write to the functional registers because it knows when CSI/VI is idle. The VI3 instead generates this syncpt when there are no pending captures and pending OP*_DONE syncpts have been sent to Host1x.
VI_CSI_PPA_LINE_START	0x04	2	Received a SOL indication on camera (CSI PPA) input stream. This SOL is detected at the VI3 input.
VI_CSI_PPA_FRAME_START	0x05	2	Valid SOF has been received by the camera (CSI PPA) input. This SOF is detected at the VI3 input.
VI_MWA_REQ_DONE	0x06	2	Predefined Operation Done for All Reqs for a Frame condition. VI3 generates this syncpt when all requests for a captured frame from camera (CSI PPA) to memory are completed.
VI_MWA_ACK_DONE	0x07	2	Predefined Operation Done for all WrAck's for a Frame condition. VI3 generates this syncpt when all WrAck for a captured frame from camera (CSI PPA) to memory are received.
VI_CSI_PPB_LINE_START	0x08	2	Received a SOL indication on camera (CSI PPB) input stream. This SOL is detected at the VI3 input.
VI_CSI_PPB_FRAME_START	0x09	2	Valid SOF has been received by the camera (CSI PPB) input. This SOF is detected at the VI3 input.
VI_MWB_REQ_DONE	0x0A	2	Predefined Operation Done for All Reqs for a Frame condition. VI3 generates this syncpt when all requests for a captured frame from camera (CSI PPB) to memory are completed.
VI_MWB_ACK_DONE	0x0B	2	Predefined Operation Done for all WrAck's for a Frame condition. VI3 generates this syncpt when all WrAck for a captured frame from camera (CSI PPB) to memory are received.
VI_CSI_PPC_LINE_START	0x0C	2	Received a SOL indication on camera (CSI1 PPA) input stream. This SOL is detected at the VI3 input.
VI_CSI_PPC_FRAME_START	0x0D	2	Valid SOF has been received by the camera (CSI1 PPA) input. This SOF is detected at the VI3 input.
VI_MWC_REQ_DONE	0x0E	2	Predefined Operation Done for All Reqs for a Frame condition. VI3 generates this syncpt when all requests for a captured frame from camera (CSI1 PPA) to memory are completed.
VI_MWC_ACK_DONE	0x0F	2	Predefined Operation Done for all WrAck's for a Frame condition. VI3 generates this syncpt when all WrAck for a captured frame from camera (CSI1 PPA) to memory are received.
VI_CSI_PPD_LINE_START	0x10	2	Received a SOL indication on camera (CSI1 PPB) input stream. This SOL is detected at the VI3 input.
VI_CSI_PPD_FRAME_START	0x11	2	Valid SOF has been received by the camera (CSI1 PPB) input. This SOF is detected at the VI3 input.
VI_MWD_REQ_DONE	0x12	2	Predefined Operation Done for All Reqs for a Frame condition. VI3 generates this syncpt when all requests for a captured frame from camera (CSI1 PPB) to memory are completed.
VI_MWD_ACK_DONE	0x13	2	Predefined Operation Done for all WrAck's for a Frame condition. VI3 generates this syncpt when all WrAck for a captured frame from camera (CSI1 PPB) to memory are received.
VI_CSI_PPE_LINE_START	0x14	2	Received a SOL indication on camera (CSI2 PPA) input stream. This SOL is detected at the VI3 input.
VI_CSI_PPE_FRAME_START	0x15	2	Valid SOF has been received by the camera (CSI2 PPA) input. This SOF is detected at the VI3 input.

Table 178: Sync Point Conditions

Condition	Value	FIFO Depth	Description
VI_MWE_REQ_DONE	0x16	2	Predefined Operation Done for All Reqs for a Frame condition. VI3 generates this syncpt when all requests for a captured frame from camera (CSI2 PPA) to memory are completed.
VI_MWE_ACK_DONE	0x17	2	Predefined Operation Done for all WrAck's for a Frame condition. VI3 generates this syncpt when all WrAck for a captured frame from camera (CSI2 PPA) to memory are received.
VI_CSI_PPF_LINE_START	0x18	2	Received a SOL indication on camera (CSI2 PPB) input stream. This SOL is detected at the VI3 input.
VI_CSI_PPF_FRAME_START	0x19	2	Valid SOF has been received by the camera (CSI2 PPB) input. This SOF is detected at the VI3 input.
VI_MWF_REQ_DONE	0x1A	2	Predefined Operation Done for All Reqs for a Frame condition. VI3 generates this syncpt when all requests for a captured frame from camera (CSI2 PPB) to memory are completed.
VI_MWF_ACK_DONE	0x1B	2	Predefined Operation Done for all WrAck's for a Frame condition. VI3 generates this syncpt when all WrAck for a captured frame from camera (CSI2 PPB)to memory are received.
VI_ISPA_DONE	0x1C	2	Predefined Operation Done condition. VI3 generates this syncpt when it finishes sending frame from the configured camera to ISPA
VI_ISPB_DONE	0x1D	2	Predefined Operation Done condition. VI3 generates this syncpt when it finishes sending frame from the configured camera to ISPB
VI_VGP0_RECD	0x1E	2	Rising edge detection on VGPx (VGPx being selected by a configuration register)
VI_VGP1_RECD	0x1F	2	Rising edge detection on VGPy (VGPy being selected by a configuration register)

VI Configuration Register Details

When the Host1x writes to any of the functional mode registers, the configuration value is immediately visible. So the software should write to such registers when the CSI/VI3 pipeline is idle. Software has visibility when the CSI/VI3 units are idle because it is not waiting for any syncpts.

When the VI3 unit receives the SOF from the CSI unit, it raises the syncpt corresponding to SOF_RECD on the appropriate syncpt index. This indicates to the software that the CSI and VI3 units are currently capturing a frame. At this time, software sends out the operational parameters for the next frame. These register writes are written to the shadow register. After the shadow register writes are complete, software indicates that the shadow registers update is done by writing to the VI_CSI_[0..5]_SINGLE_SHOT_CAPTURE register. After the single shot capture trigger is set, all shadow copies are now programmed and the VI3 unit will wait for the next EOF for transferring the active copy. When it receives the EOF and the previous frame is error free and correct, it updates the active registers from the shadow copy.

The VI3 unit marks all writes to memory as writes with ACK required. It also maintains a counter to keep track of number of writes outstanding. When the EOF is received from the CSI and there are no WrAcks pending, the VI3 unit generates MW[A..F]_REQ_DONE to Host1x. Software uses this syncpt to pass this memory buffer to the next unit which will process the frame. In case the VI3 unit is configured to send the frame to ISP, it raises the ISPA_DONE or ISPB_DONE syncpt when the EOF packet is sent to the ISP. Depending on the configuration of HOST1X_UPDATE register configuration, these syncpts are either raised when the frame is known to be good or when any frame is sent to memory/ISP.

31.2.4.2 VI-I2C Dedicated Host1x Interface

The Host1x interface is the primary method of communicating the control state from the CPU to the VI_I2C. It is responsible for the register write and register read requests. The VI_I2C shares the 4 channels allocated to the VI3 unit and for the Host1x interface. The use of two channels from the available four allows the VI_ I2C block to the software to simultaneously capture images from 2 cameras or devices asynchronous to each other.

VI-I2C Sync Points

The VI_I2C provides a similar set of sync points as seen with the VI for managing transactions based on frame events. Since the VI_I2C does not require shadowing, there is no need to provide for a FIFO to buffer the sync point. The SOF, SOL, and EOF sync points are available to enable the ability to manage the direct transaction and to provide general program control. The following table describes the sync points for the VI_I2C block.

Table 179: VI_I2C Sync Point Conditions

Condition	Value	Description
IMMEDIATE	0x00	Predefined Immediate condition. The VI3 unit returns this syncpt immediately.
OP_DONE	0x01	Predefined Operation Done Condition
RD_DONE	0x02	Predefined Read Done condition. The VI3 unit treats this as IMMEDIATE because CSI/VI do not have any read ports.
REG_WR_SAFE	0x03	Predefined Register Write Safe condition. Technically, software should know when it is safe to write to the functional registers as it knows when CSI/VI is idle. The VI3 unit instead generates this syncpt when there are no pending captures and pending OP*_DONE syncpt have been sent to Host1x
VI_I2C_ERECOVERY_TX_DONE	0x04	An Error Recovery path Direct packet TX has completed.
VI_I2C_DIRECT_TX_DONE	0x05	A Direct mode packet TX transaction has completed
VI_I2C_DIRECT_RX_DONE	0x08	A Direct mode packet RX transaction has completed
VI_I2C_CSI_PPA_FS	0x09	Valid SOF has been received by the camera (CSI PPA) input. This SOF is detected at the VI3 input.
VI_I2C_CSI_PPB_FS	0x0A	Valid SOF has been received by the camera (CSI PPB) input. This SOF is detected at the VI3 input.
VI_I2C_CSI_PPA_FE	0x0B	Valid EOF has been received by the camera (CSI PPA). This is detected at the VI3 input.
VI_I2C_CSI_PPB_FE	0x0C	Valid EOF has been received by the camera (CSI PPB). This is detected at the VI3 input.
VI_I2C_CSI_PPA_LS	0x0D	Received an SOL indication on the camera (CSI PPA) input stream. This SOL is detected at the VI3 input.
VI_I2C_CSI_PPB_LS	0x0E	Received an SOL indication on the camera (CSI PPB) input stream. This SOL is detected at the VI3 input.
VI_I2C_CSI1_PPA_FS	0x0F	Valid SOF has been received by the camera (CSI1 PPA) input. This SOF is detected at the VI3 input.
VI_I2C_CSI1_PPB_FS	0x10	Valid SOF has been received by the camera (CSI1 PPB) input. This SOF is detected at the VI3 input.
VI_I2C_CSI1_PPA_FE	0x11	Valid EOF has been received by the camera (CSI1 PPA). This is detected at the VI3 input.
VI_I2C_CSI1_PPB_FE	0x12	Valid EOF has been received by the camera (CSI1 PPB). This is detected at the VI3 input.
VI_I2C_CSI1_PPA_LS	0x13	Received an SOL indication on the camera (CSI1 PPA) input stream. This SOL is detected at the VI3 input.
VI_I2C_CSI1_PPB_LS	0x14	Received an SOL indication on the camera (CSI1 PPB) input stream. This SOL is detected at the VI3 input.
VI_I2C_CSI2_PPA_FS	0x15	Valid SOF has been received by the camera (CSI2 PPA) input. This SOF is detected at the VI3 input.
VI_I2C_CSI2_PPB_FS	0x16	Valid SOF has been received by the camera (CSI2 PPB) input. This SOF is detected at the VI3 input.
VI_I2C_CSI2_PPA_FE	0x17	Valid EOF has been received by the camera (CSI2 PPA). This is detected at the VI3 input.
VI_I2C_CSI2_PPB_FE	0x18	Valid EOF has been received by the camera (CSI2 PPB). This is detected at the VI3 input.
VI_I2C_CSI2_PPA_LS	0x19	Received a SOL indication on the camera (CSI2 PPA) input stream. This SOL is detected at the VI3 input.
VI_I2C_CSI2_PPB_LS	0x1A	Received a SOL indication on the camera (CSI2 PPB) input stream. This SOL is detected at the VI3 input.

VI I2C Configuration Register Details

When the VI_I2C Host1x writes to any of the functional mode registers, the configuration value are immediately visible.

31.2.5 VI I2C Clocking and Reset

The VI_I2C block in the VI3 unit is clocked independent of the VI3. In order to enable the VI_I2C along with its embedded I2C controller, the clocks to both must be enabled. The VI_I2C clock is enabled by setting the CLK_ENB_VI_I2C bit in the CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0 register and the clock for the embedded I2C controller by setting the CLK_ENB_I2C_SLOW bit in the CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0 in the CAR block. In order to program the VI_I2C, the Host1x must also be enabled.

The VI_I2C block and the VI_I2C dedicated Host1x interface have independent resets from the VI3. The VI_I2C portion of the VI3 is reset by setting the SWR_VI_I2C_RST in the CLK_RST_CONTROLLER_RST_DEVICES_Y_0 register in the CAR block. This resets everything associated with the VI_I2C block including the I2C Host1x slave interface, the I2C IP block, and all VI_I2C control logic.

31.2.6 Error Status and Interrupts

This section discusses the reporting of various error conditions that can occur with the VI3 unit during the operation of the camera subsystem. The VI and the VI_I2C have different requirements for reporting various error conditions.

31.2.6.1 VI Error Status and Interrupts

The VI_CSI_[0..5]_ERROR_STATUS and VI_CSI_[0..5]_ERROR_INT_MASK functional registers are expected to be active for the camera use session. The remaining VI3 functional registers are described in the following table.

The VI_CSI_[0..5]_ERROR STATUS is a read-only register that indicates whether an error has occurred. All of the conditions defined in this register can be programmed to generate an interrupt using the VI_CSI_[0..5]_ERROR_INT_MASK as described in the following table.

Table 180: VI Error Status and Interrupt Mask Registers Functional Register Definitions

Register	Field	Bits	Reset Value	Description
VI_CSI_[0..5]_ERROR_STATUS	LINE_WIDTH_SHORT_ERROR:	0	X	Flagged if the frame line width is smaller than WIDTH. Write 1 to clear.
	LINE_WIDTH_LONG_ERROR:	1	X	Flagged if the frame line width is larger than WIDTH. Write 1 to clear
	FRAME_HEIGHT_SHORT_ERROR	2	X	Flagged if the frame height is smaller than HEIGHT. Write 1 to clear
	FRAME_HEIGHT_LONG_ERROR	3	X	Flagged if the frame height is larger than HEIGHT. Write 1 to clear
	CSI_FRAME_ERROR	4	X	Flagged if the EOF field from the CSI has FRAME_ERROR bit set
	WATCHDOG_INT	5	X	This event occurs when Watchdog timer expires and the EOF is not received on the CSI2VI interface
VI_CSI_[0..5]_ERROR_INT_MASK	LINE_WIDTH_SHORT_INT_MASK	0	0x0	Line width short error interrupt mask
	LINE_WIDTH_LONG_INT_MASK	1	0x0	Line width long error interrupt mask
	FRAME_HEIGHT_SHORT_INT_MASK	2	0x0	Frame height short error interrupt mask
	FRAME_HEIGHT_LONG_INT_MASK	3	0x0	Frame height long error interrupt mask
	CSI_FRAME_ERROR_INT_MASK	4	0x0	CSI error interrupt mask
	WATCHDOG_INT_MASK:	5		Watchdog Timer 0 Interrupt Mask. This controls interrupts when a Watchdog trigger event occurs for the CSI stream. 0 = Disabled 1 = Enabled

Watchdog Counter

To detect catastrophic error conditions in which the CSI or VI3 unit has ceased operating, there are six Watchdog timers provided with the VI3 unit. The WD timers are associated with the image data streams being presented to the VI3 through the CSI, CSI1, and CSI2 PPA and PPB ports. Once the WD timer is configured for an image data stream, the timer must be prevented from expiring by the hardware. If the hardware does not disable the timer before the count expires, an interrupt is generated.

The WD timer counter is initialized to the value in the PERIOD register on the arrival of the SOF packet at the CSI to the VI3 interface for the active stream. The counter then proceeds to count down one count for every pixel clock. If the count reaches zero, with the WATCHDOG_INT_MASK bit set in the VI_CSI_[0..1]_ERROR_INT_MASK register, an interrupt is generated.

If an EOF packet is received at the CSI port, the countdown is stopped. In normal operation, this prevents the counter from reaching zero and generating an interrupt.

The ISP also provides a WD timer system.

31.2.6.2 VI_I2C Status and Interrupts

This section discusses the reporting of various status and error conditions that can occur with the VI_I2C during the operation of the I^2C controller. Refer to Chapter 35: I2C Controller in this TRM for details on the I^2C registers. The corresponding VI3 I^2C registers and I^2C reference registers are defined at the end of this chapter and summarized in the table below.

Table 181: I^2C Status and Interrupt Mask Registers Fields Used for Master Transactions

VI_I2C Virtual Status and Interrupt Reference Registers	I2C Controller Mapped Register
VI_ I2C_STREAM_DIRECT_I2C_INTERRUPT_MASK_REGISTER_0 VI_ I2C_STREAM_ERECOVERY_I2C_INTERRUPT_MASK_REGISTER_0	I2C_INTERRUPT_MASK_REGISTER_0
VI_ I2C_ STREAM_DIRECT_I2C_INTERRUPT_SET_REGISTER_0 VI_ I2C_ STREAM_ERECOVERY_I2C_INTERRUPT_SET_REGISTER_0	I2C_INTERRUPT_SET_REGISTER_0
VI_ I2C_ STREAM_DIRECT_I2C_INTERRUPT_STATUS_REGISTER_0 VI_ I2C_ STREAM_ERECOVERY_I2C_INTERRUPT_STATUS_REGISTER_0	I2C_INTERRUPT_STATUS_REGISTER_0
VI_ I2C_ STREAM_DIRECT_I2C_INTERRUPT_SOURCE_REGISTER_0 VI_ I2C_ STREAM_ERECOVERY_I2C_INTERRUPT_SOURCE_REGISTER_0	I2C_INTERRUPT_SOURCE_REGISTER_0
VI_I2C_ STREAM_DIRECT_I2C_PACKET_TRANSFER_STATUS_0 VI_I2C_ STREAM_ERECOVERY_I2C_PACKET_TRANSFER_STATUS_0	I2C_PACKET_TRANSFER_STATUS_0
VI_I2C_ STREAM_DIRECT_I2C_ BUS_CLEAR _STATUS_0 VI_I2C_ STREAM_ERECOVERY_I2C_BUS_CLEAR _STATUS_0	I2C_BUS_CLEAR_STATUS
VI_I2C_ STREAM_DIRECT_I2C_FIFO_STATUS_0 VI_I2C_ STREAM_ERECOVERY_I2C_FIFO_STATUS_0	I2C_FIFO_STATUS_0

The interrupt registers are set up to generate interrupts based on various error conditions or to indicate a packet transfer has completed. Since the VI_I2C is only supporting master packet transactions and provides sync points for indicating successful completion of a packet transaction, only the following subset of the available interrupts is needed:

- BUS_CLEAR_DONE
- TLOW_MEXT_TIMEOUT_EN
- TLOW_SEXT_TIMEOUT_EN
- TIMEOUT_INT_EN
- TFIFO_OVF_INT_EN
- RFIFO_UNF_INT_EN
- NOACK_INT_EN
- ARB_LOST_INT_EN

The status and interrupt registers are accessible by software and can be read at any time to get the status of the current transaction. In the case of an interrupt, software can access the status registers and identify the packet ID which was in process when an error occurred. All I^2C interrupt registers are accessed through ERECOVERY or DIRECT FIFO.

The VI_I2C hardware has access to the interrupt status registers signals during normal operation and uses the packet transaction complete into an increment for a particular transaction complete sync point and to manage the data flow to the TX_FIFO by using the DMA REQ signal. In order to facilitate the VI_I2C management of the TX_FIFO and sync points, the I^2C Controller provides the interrupt signals as the interface to the VI_I2C.

In addition to the interrupts managed within the I^2C controller, it is necessary to provide some additional indication to software if the DIRECT FIFO has overflowed. The combined I^2C and VI_I2C interrupt status is provided in the VII2C_INT_STATUS register as described in the VI I^2C Registers section.

Although the slave interrupts are provided in the VII2C_INT_STATUS register, these signals are not used in the VI_I2C.

In addition to the interrupt status and mask registers, the VII2C_EROVERY_MASK register is provided to manage the passing of the arbitration to the Error Recovery path when an interrupt is generated. This register should be programmed with the applicable bits set, which are used to generate an interrupt only. The Error Recovery path will gain immediate access to the APB bus when any of the programmed error conditions occur.

31.2.7 VI3 Programming Guidelines

The programming of the VI3 unit to initiate data capture involves setting up the image sensor, the CSI unit, the VI unit, and the ISP unit if the data is targeted for processing by the ISP. There are several different modes of image capture supported by the camera subsystem including progressive frame capture, interlaced frame capture, and interleaved frame capture. The capture modes may be set up as either single or dual stream to support the various user models, with the exception of the interleaved still capture. In the interleaved capture mode, both streams are assumed to be active.

31.2.7.1 Basic Progressive Frame Capture Sequence

The basic progressive frame capture programming sequence supports a number of user models involving both single and dual stream image captures. The basic sequence for the progressive capture is as follows:

1. Program the sensor or sensors in the dual stream use case.
2. Program the CSI and VI with parameters that do not change every frame.
3. Program Frame #1 specific parameters for the CSI and VI.
4. Enable CSI pixel parsers after completing PPA and/or PPB setup.
5. Arm the syncpt.
6. Trigger single shot by setting CAPTURE in the VI_CSI_[0..1]_SINGLE_SHOT register

   If (Outstanding capture requests == 1)

   Wait for SOF

   Else if (Outstanding capture requests > 1)

   Wait for MW_REQ_DONE
7. Program Frame #2 specific parameters for CSI and VI.
8. Repeat from step 5 for the next frames.

31.2.7.2 Basic Interlaced Frame Capture Sequence

The interlaced capture programming sequence is similar to the basic progressive capture sequence. However, since the interlaced capture results in the capture of a top and bottom field pair for each frame, there are some differences in how the hardware manages the syncpt generation and the single shot generation. Because the frame capture consists of a frame pair, both the syncpt and single shot are also managed based on frame pairs. The basic sequence for the interlaced capture is as follows:

1. Program the sensor or sensors in dual stream use case
2. Program the CSI and VI with parameters that do not change every frame.
3. Program Frame #1 Top/Bottom Field specific parameters for the CSI and VI.
4. Program the CSI TOP_FIELD_FRAME and TOP_FIELD_FRAME_MASK
5. Enable CSI Pixel Parser after completing the PPA and/or PPB setup
6. Arm syncpts for the field pair capture
7. Trigger single shot by setting CAPTURE in the VI_CSI_[0..5]_SINGLE_SHOT register

   If (Outstanding capture requests == 1)

   Wait for PP[A..B]_ FRAME_START

   Else if (Outstanding capture requests > 1)

   Wait for MW_REQ_DONE
8. Program Frame #2 specific parameters for CSI and VI.
9. Repeat from step 6 for next frames.

Step 4 is unique to the interlaced capture and is necessary for generating the top/bottom field signaling for the VI3. This signaling manages the data flow to memory and handling of field drop error conditions. The SOF syncpt is generated only when a top field is signaled on the CSI to VI interface, and the MW_REQ_DONE syncpt is generated only after a top and bottom field are paired and written to memory. The VI3 unit generates a single shot trigger for the top and bottom field when the interlaced mode is enabled. In addition, the VI3 unit provides for a frame-dropping mechanism which automatically re-issues a single shot to capture another field if a frame is dropped.

31.2.8 VI-I^{2}C Programming Guidelines

This section describes the programming sequences for the VI-I^{2}C block when initializing and controlling multiple devices within the camera subsystem. In the following sections, several scenarios are approached.

31.2.8.1 Register Programming Overview

This subsection describes some of the registers associated with setting up and programming the various modes supported with the VI-I^{2}C.

I^{2}C Frequency Divisor and Interface Timing Register

The frequency divisor and interface timing registers are used to set up the Standard/HS mode clock frequency and for tuning the I^{2}C interface bus timing, if needed. Separate bit fields are provided for non-HS (STD/FM/FM+) and HS modes timing:

- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_CLK_DIVISOR_REGISTER_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_INTERFACE_TIMING_0_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_INTERFACE_TIMING_1_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_HS_INTERFACE_TIMING_0_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_HS_INTERFACE_TIMING_1_0

These functional registers are not expected to be updated during a camera use session. During the initialization of the VI_I2C controller, the timing is configured prior to initiating any command transactions to external devices. Refer to Chapter 35: I2C Controller in this TRM for detailed information on programming these bit fields.

I^{2}C Configuration and Configuration Load Register

The I^{2}C configuration and configuration load registers are used to set up the I^{2}C master configurations and to initiate transfer of the software programmed configuration in the I^{2}C registers to hardware registers that are used by the actual logic.

DIRECT MODE:

- VI_I2C_STREAM_DIRECT_I2C_CNFG_0
- VI_I2C_STREAM_DIRECT_I2C_CONFIG_LOAD_0
- VI_I2C_STREAM_ERECOVERY_I2C_CNFG_0
- VI_I2C_STREAM_ERECOVERY_I2C_CONFIG_LOAD_0

The VI_I2C_CONFIG register is used to both configure the I^2C master and to initiate transactions on the bus. It contains a number of fields which allow software to program the number of bytes, single or two slave transaction, and whether to send a start byte or not. The bit VI_I2C_CNGF [9] is used to issue a “Begin-Transaction” command to the VI-I^2C Master Controller when in normal mode and VI_I2C_CNFG[10] is used to initiate transfer in packet mode. This bit is reset by hardware and other bits of the register are masked for writes when this bit is programmed to one. Hence, the firmware should first configure all other registers and the bits [8:0] of I2C_CNFG register before the bit I2C_CNFG [9] is programmed to one when in direct mode. Similarly in the synchronized mode all of the registers should be first presented to the VI-I^2C controller with these bit fields zero, and HW will re-write the register with the “GO’ bit set to one to initiate transactions.

This CONFIG load register is used to transfer the software programmed configuration in I^2C registers to hardware internal registers that are used in the actual logic. The CONFIG load register has three bit fields:

- MSTR_CONFIG_LOAD for regular master and Bus Clear master logic
- SLV_CONFIG_LOAD bit field for slave controller logic
- TIMEOUT_CONFIG_LOAD bit field for SMBUS timeout logic

From the programming side, software has to set them to 1 for the actual register configuration to take effect. It is as if software is programming only shadow registers through regular configuration. When these load_config bit fields are set to 1, the regular/ shadows registers configuration is transferred to the hardware internal active registers. So software has to set these bit fields at the end of all regular registers configuration. But in normal or non-packet mode, these should be set to 1 before the I2C_I2C_CNFG_0[SEND] bit is set to 1. The I2C_I2C_CNFG_0[SEND] bit triggers a transfer on the I^2C bus and should be programmed at the end of entire configuration (these config_load bits are hardware auto-clear bits). Hardware clears these bit fields once the register configuration is moved to hardware internal active registers. So software has to wait until these bits are auto-cleared before there is any further programming.

Because the VI_I2C supports the master mode of operation only, the master and timeout bit fields must be programmed. For the MSTR_CONFIG_LOAD and TIMEOUT_CONFIG_LOAD, the following VI-I^2C Master Configuration Load registers are loaded:

DIRECT MODE:

- VI_I2C_STREAM_DIRECT_I2C_CNFG_0
- VI_I2C_STREAM_ERECOVERY_I2C_CNFG_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_CLK_DIVISOR_REGISTER_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_BUS_CLEAR_CONFIG_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_INTERFACE_TIMING_0_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_INTERFACE_TIMING_1_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_HS_INTERFACE_TIMING_0_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_HS_INTERFACE_TIMING_1_0
- VI_I2C_STREAM_[DIRECT,ERECOVERY]_I2C_TLOW_SEXT_0

31.2.8.2 VI_I2C Direct Mode Programming

The direct mode programming provides for either device initialization or provides a means to issue transactions during frame capture.

1. Program the functional registers:
   - VI_I2C_STREAM_DIRECT_I2C_CLKEN_OVERRIDE_0
   - VI_I2C_STREAM_DIRECT_I2C_CLK_DIVISOR_REGISTER_0
   - VI_I2C_STREAM_DIRECT_I2C_INTERFACE_TIMING_0_0
   - VI_I2C_STREAM_DIRECT_I2C_INTERFACE_TIMING_1_0
   - VI_I2C_STREAM_DIRECT_I2C_HS_INTERFACE_TIMING_0_0
   - VI_I2C_STREAM_DIRECT_I2C_HS_INTERFACE_TIMING_1_0
   - VI_I2C_STREAM_DIRECT_I2C_BUS_CLEAR_CONFIG_0
   - VI_I2C_STREAM_DIRECT_I2C_TLOW_SEXT_0
2. Set up Interrupt MASK and STATUS registers
3. Arm DIRECT_PCKT_DONE syncpt
4. Program VI_I2C_STREAM_DIRECT_I2C_CONFIG_0, bit 10, PACKET_MODE_EN = 1
5. Program VI_I2C_STREAM_DIRECT_I2C_FIFO_CONTROL trigger level
6. Program VI_I2C_ STREAM_DIRECT_I2C_CONFIG_LOAD
7. Write VI_I2C_ STREAM_DIRECT_I2C_TX_PACKET_FIFO[0..2] with generic header, payload size, and I2C specific header
8. Write VI_I2C_ STREAM_DIRECT_I2C_TX_PACKET_FIFO[2..N] with payload or additional headers plus payload.
9. Write VI_I2C_STREAM_DIRECT_STREAM_DIRECT_FENCE with frame number.

   Wait for VI_I2C: PCKT DONE syncpt
10. Read VI_I2C_ STREAM_DIRECT_I2C_RX_FIFO if read transaction.
11. Repeat steps 2 through 12 for next transaction.

31.2.8.3 VI_I2C MMIO Direct Mode Programming

The programming of the VI_I2C may also be done using the MMIO approach as an alternative to the standard Host1x programming model. In this case, the register writes are done using the Linux kernel APIs. A basic pseudo-code example using writel, readl and nvhost_syncpt_* are Linux kernel APIs is illustrated below.

```
writel(val, reg-address);
writel(val, reg-address);
writel(val, reg-address);
writel(val, reg-address);
writel(val, 0x0);
SyncPt-val = nvhost_syncpt_incr_max();
nvhost_syncpt_wait(SyncPt-val);
```

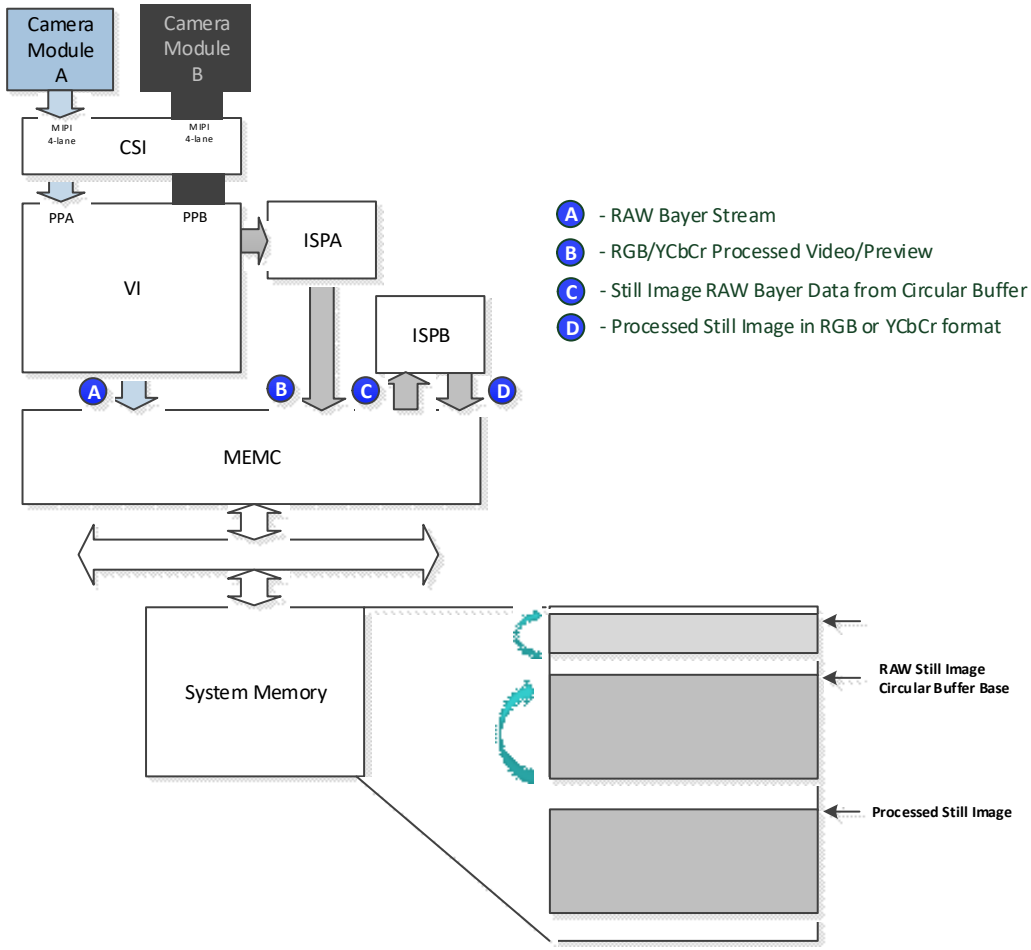
31.2.9 Frame Capture Modes

There are several frame capture scenarios available within the camera subsystem, which utilize various resources within the VI3 unit. The modes include scene preview or simple progressive video capture, multi camera progressive or interlaced video capture, high-resolution still capture, high-resolution zero shutter lag still capture, and an 8-lane interleaved capture.

31.2.9.1 High Resolution and ZSL Still Image Capture

The high resolution and ZSL still capture models utilize the VI3 resources for providing a path to system memory for Raw Bayer image data while simultaneously maintaining an image data stream to the ISP(s) for preview. The following figure shows the active data paths during mono ZSL capture mode.

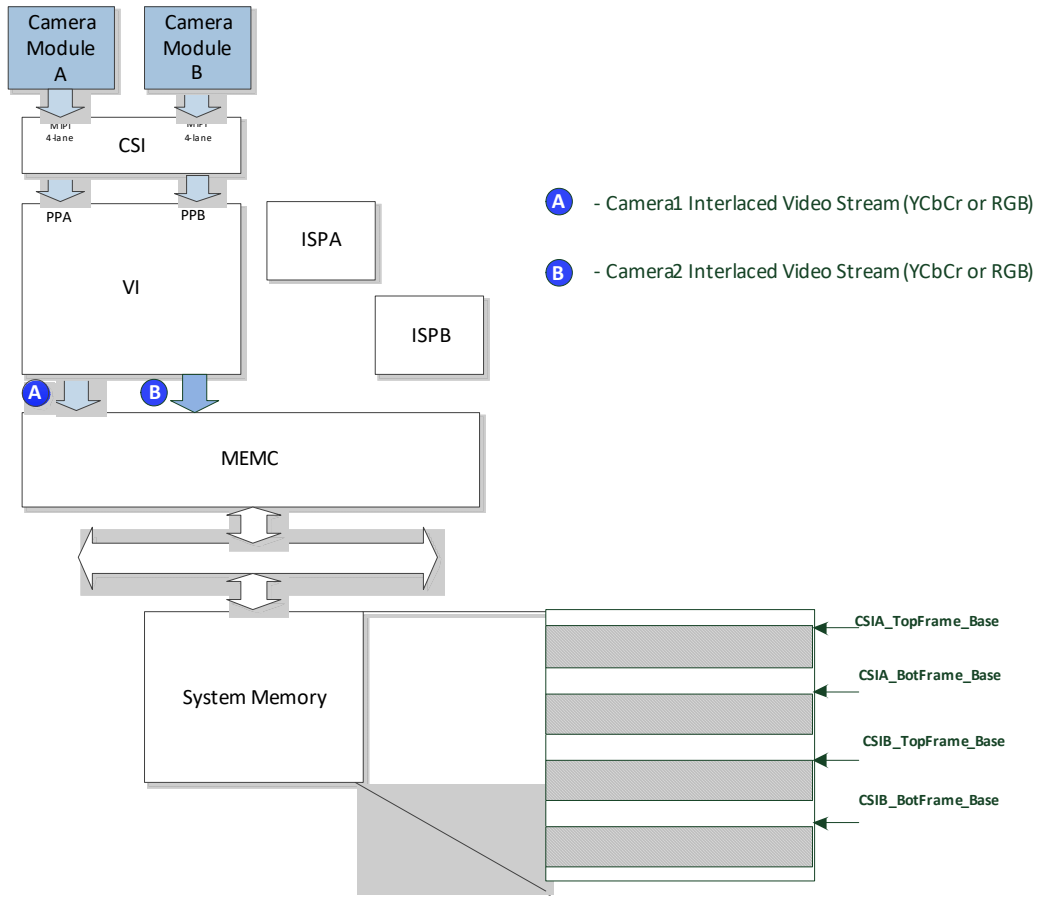
Figure 128: Zero Shutter Lag Mode



31.2.9.2 Interlaced Video Capture

The interlaced capture model provides for the management of interlaced video from an external camera module. The interleaved image data is presented to the CSI unit in a sequence of odd and even frames which represent the top and bottom fields in the interleaved composite frame. Since the interleaved image frames are not targeting raw Bayer processing by the ISP, the data formats are limited to either be RGB or YCbCr. The VI3 capture of interlaced video allows up to two camera pixel streams and associated channel pointers frame pointers to be managed simultaneously. The interlaced video capture mode is shown in the following figure.

Figure 129: Interlaced Capture Mode



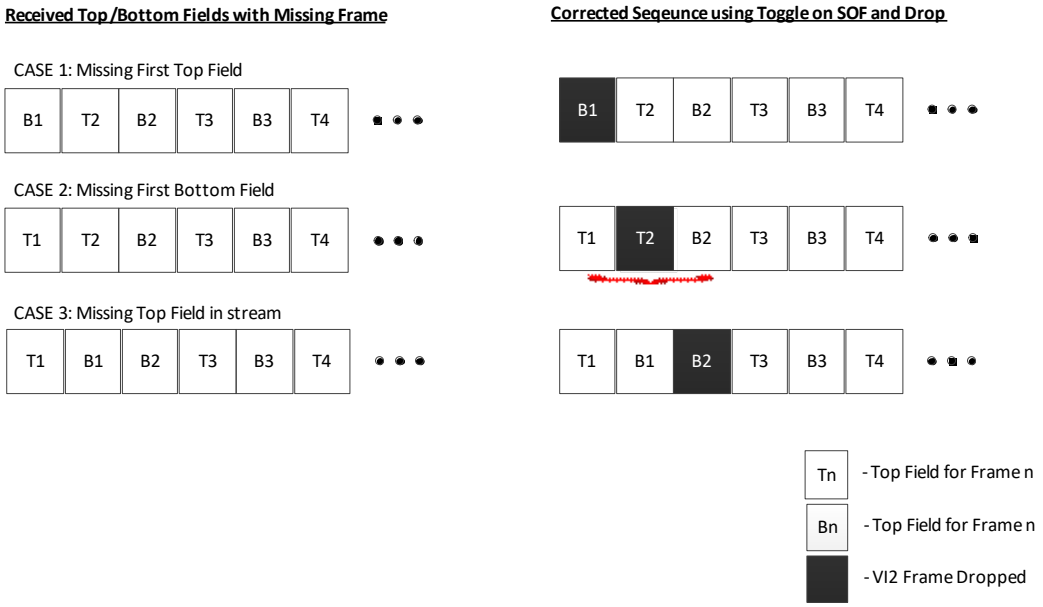
Twelve (12) pointers are available for managing the interleaved image streams. Three are available for the top field and three are available for the bottom fields for each of the two streams. All three pointers are used for each stream when the data is YCbCr422 and the target is planar, otherwise either two or only one are needed for semi-planar and non-planar formats, respectively.

The CSI unit signals the VI3 unit when a frame is either the top or bottom field by setting or clearing the LSB on the CSI[A,B]2VI_DATA, CSI1[A,B]2VI_DATA, or CSI2[A,B]2VI_DATA bus when the SOF is being sent. The VI3 unit uses the signaling to provide for switching between the top and bottom field source and stride registers setting used for the address calculations. The signals are managed through the CSI by programming the appropriate CSI, CSI1, or CSI2 instance's CSI_PPA_TOP_FIELD_FRAME_MASK and CSI_PPA_TOP_FIELD_FRAME parameters. The actual frame number used in the associated calculation is taken from the WC field of the Frame Start short packet.

Interlaced Capture Error Handling

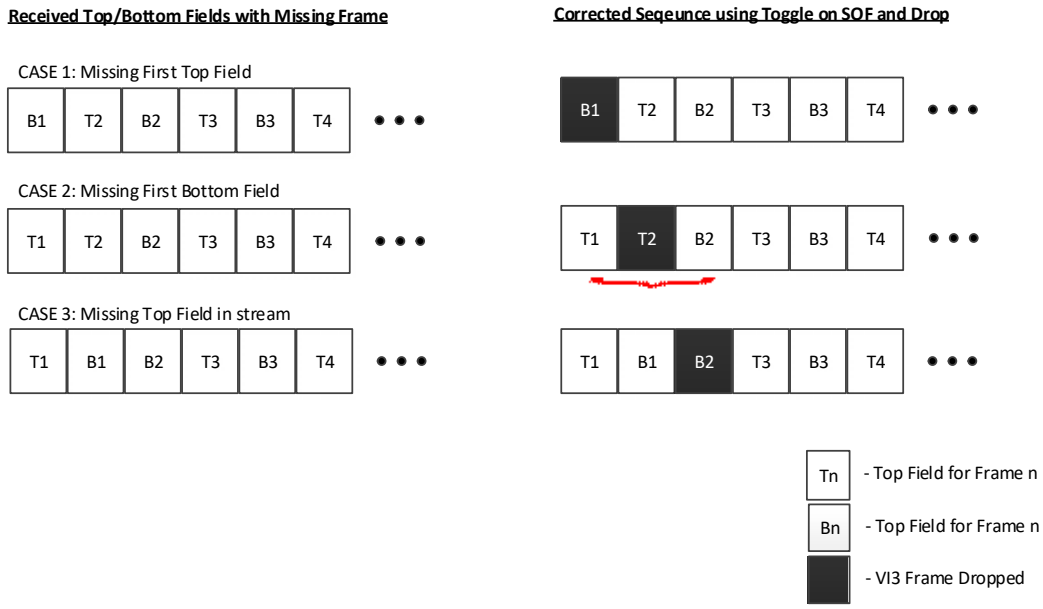
The interlaced capture generates two frames of data, one for the top field and one for the bottom field. It is important that the pairing of the frames in the top and bottom field buffers is accurate since any mismatch could result in artifacts that could propagate as errors in future frames captured in the video sequence. The issue is manifested with the occurrence of dropped frames either at the camera module or in the CSI unit. If a frame is lost, several pairing errors are possible.

Figure 130: Interleaved Field Pairing Errors due to Dropped Frames



In order to manage the dropped frame event, a flag is maintained in hardware that toggles on every SOF received from the CSI while in interlaced video mode. This indicates whether the frame is even or odd. The VI3 unit also compares the state of the TOP field bit during the SOF. If the flag is odd and the top field bit is not set, the received field is then dropped. In a similar fashion, a missing bottom field results in a top field being dropped. The toggle and drop correction are illustrated in the following figure.

Figure 131: Error Correction for Interlaced Capture



The loss of a bottom field results in a single pairing error but is not propagated into the following field pair. It is considered to be an acceptable error tradeoff.

Interlaced Single Shot and Sync Points

The programming sequence for the interlaced capture is basically the same as for the progressive capture. The primary difference between the two modes is that the interlaced mode operates on the concept of top and field pairs. The VI3 unit manages the frame single shot trigger and EOF events based on the successful pairing of top and bottom fields.

The VI_CSI_[0..5]_SINGLE_SHOT is programmed once for each top and bottom field pair in the interlaced mode of frame capture and hardware manages the generation of a single shot trigger for both the top and bottom field capture. In the event that a top field is not signaled by the CSI following the issue of the single shot capture, the frame is dropped and the single shot is re-issued until a top field is indicated. Once a top field has been successfully captured and an EOF received, the VI3 unit then issues a single shot capture for the bottom field. In a similar fashion with a missing top field first, the single shot is re-issued if a bottom filed is not signaled after the top field was successfully captured.

Sync points are also managed in a field pair fashion making it necessary to arm a sync point once for the frame event. For example, the VI_CSI_PP[A..F]_FRAME_START sync points are only returned on a successful start of a top field capture and similarly the VI_MW[A..F]_REQ_DONE sync points are only returned after the successful completion of a bottom field capture in a top/bottom field pair.

31.2.9.3 Stereo Still and Video Capture

The stereo still and video capture modes can utilize ISPA and ISPB to process the left and right camera streams simultaneously in support of preview, video capture, and the combination of preview and high-resolution still processing. The stereo capture models should be managed carefully to achieve maximum throughput and minimize system loading. The stereo capture modes are listed in the following table.

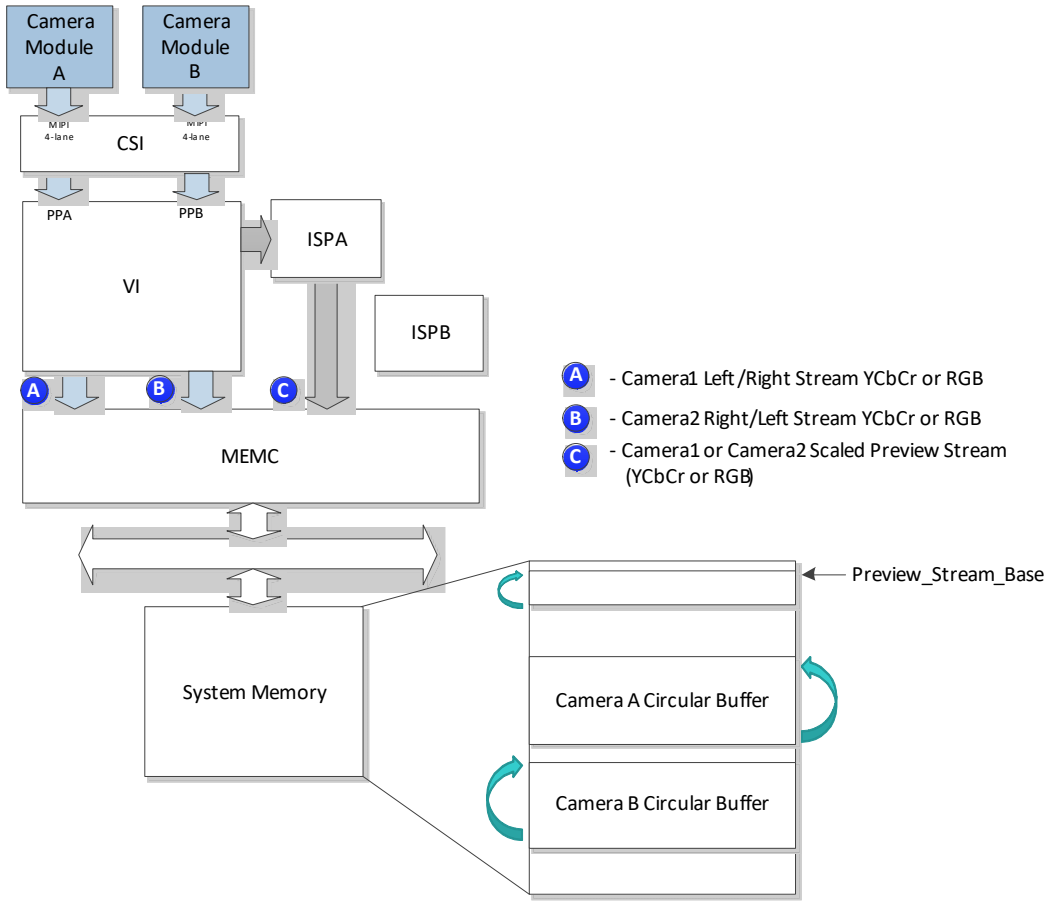
Table 182: Stereo Image/Video Capture Modes

Stereo Mode	Data Type	
Stereo Preview	YUV/RGB/RAW	Both Streams Active; ISPA or ISPB may be used for preview management; Should provide L/R preview switching; Raw data can be processed through the ISP or YUV. RGB data may be enhanced or delivered to memory through the MCCIF.
Stereo Preview for ZSL Still	YUV/RGB	Both Streams Active; ISPA or ISPB may be used for preview management; Should provide L/R preview switching; High resolution stills are delivered to the circular buffer in system memory through the MCCIF
	RAW	Both Streams Active; ISPA or ISPB used for Raw processing of one stream. Should provide L/R preview switching. High resolution Raw stills are delivered to the circular buffer in system memory through the MCCIF. During capture, the idle ISP may be used for processing so that preview can be continuous using the other ISP.
Stereo Preview + Video Capture	YUV/RGB/RAW	Both Streams Active; ISPA or ISPB may be used for preview management. Should provide L/R preview switching. Raw data may be processed through the ISP or YUV. RGB data may be enhanced or delivered to memory through the MCCIF.

There are three basic modes of operation when engaging the stereo capture user model: the preview of the image data for scene composition, the capture of a high resolution stereo pairs, and the capture of stereo video.

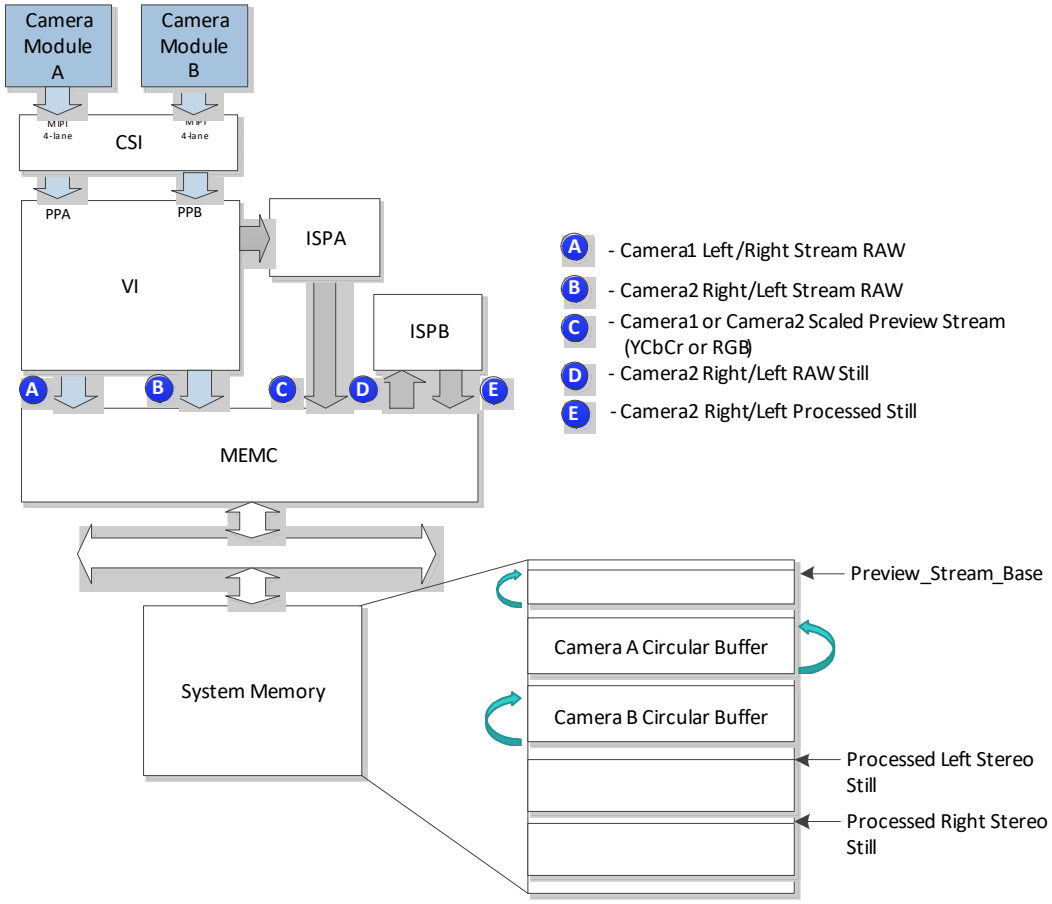
The still and video capture models for YUV or RGB processed data from the camera module may utilize the ISPA or ISPB resource as a scaling and image enhancement engine by providing the image data from either the left or right camera directly to the ISP. In most cases, the capture of processed data in a YUV or RGB format will be written directed to memory where it can be accessed by various resources in the system. The requirement for different resolutions for LCD display and the frame capture makes this option attractive since it can help to minimize memory traffic. A ZSL stereo capture sequence where both high-resolution streams are being stored in separate circular buffers for the left and right channels is illustrated in the following figure.

Figure 132: Zero Shutter Stereo Capture for YUV or RGB Image Data



The still and video capture models for Raw data utilize both ISPA and ISPB simultaneously under most conditions. During the stereo video capture, the image streams being captured by the left and right camera modules can each be assigned to either ISPA or ISPB, allowing a preview to be extracted from either processed stream with the stereo video being available to the compression resources. The ZSL stereo capture model also utilizes both ISPA and ISPB, however, only one ISP is used for managing the preview, and the second is idle until a still capture is initiated. Both high-resolution Raw streams are provided to system memory through the MCCIF to maintain the circular buffer for the left and right channels. Once the ZSL still capture is initiated, the idle ISP can be utilized to process the still images while the other is available to maintain a preview and allow a new ZSL sequence to be initiated with little delay. The basic sequence for the Raw stereo capture is shown in the following figure.

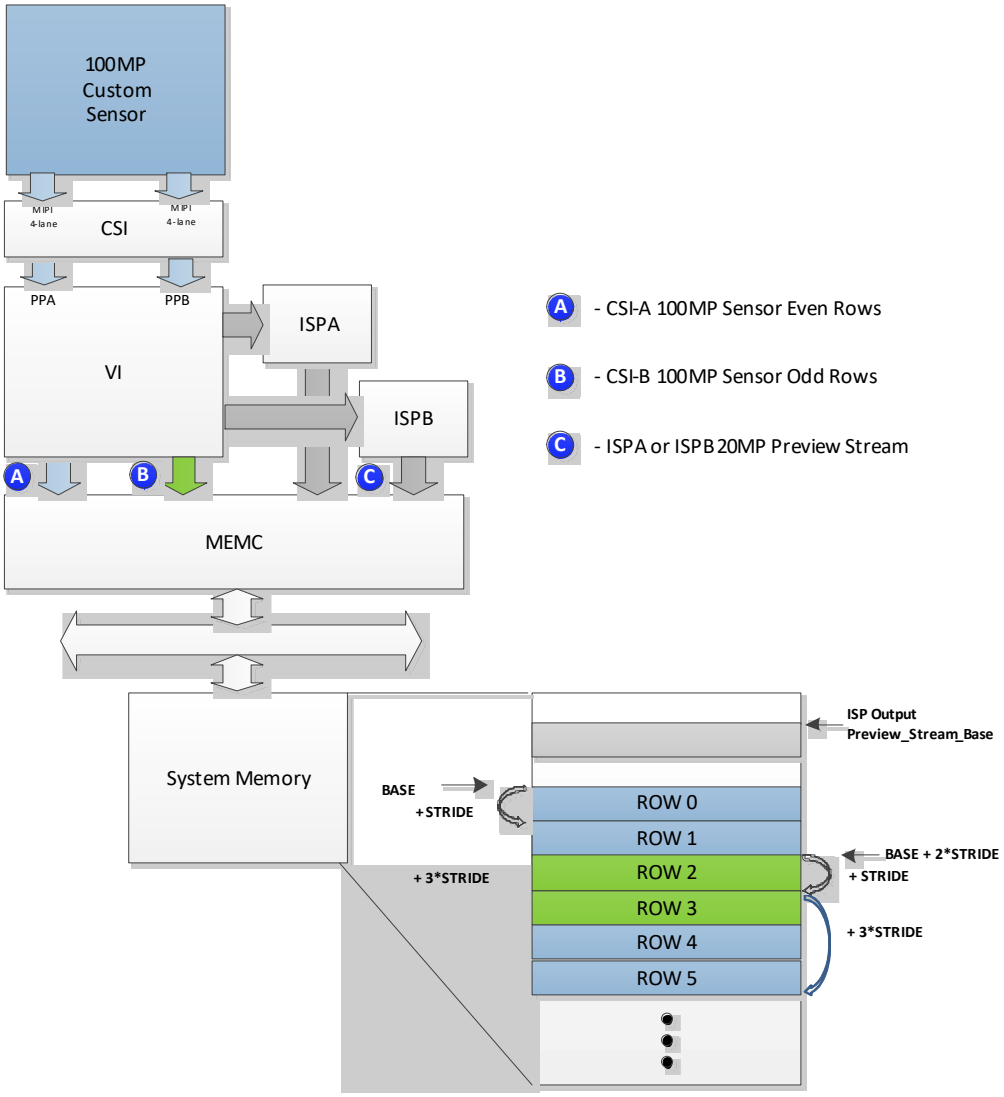
Figure 133: Zero Shutter Stereo Capture for Raw Image Data



31.2.9.4 Interleaved Capture

The interleaved capture mode supports management of pixel data delivered using all 8 available MIPI lanes. In this mode of operation, image data is delivered across all 8 lanes with two rows being provided simultaneously to two of the CSI 4-lane MIPI ports. In this mode, the VI3 unit manages the interleaving of the even and odd rows when writing the data to system memory. The sensor interfaces to the CSI unit through two of the 4-lane MIPI bricks using all four lanes for each interface. The following figure illustrates the interleaved capture model.

Figure 134: Interleaved Capture Model

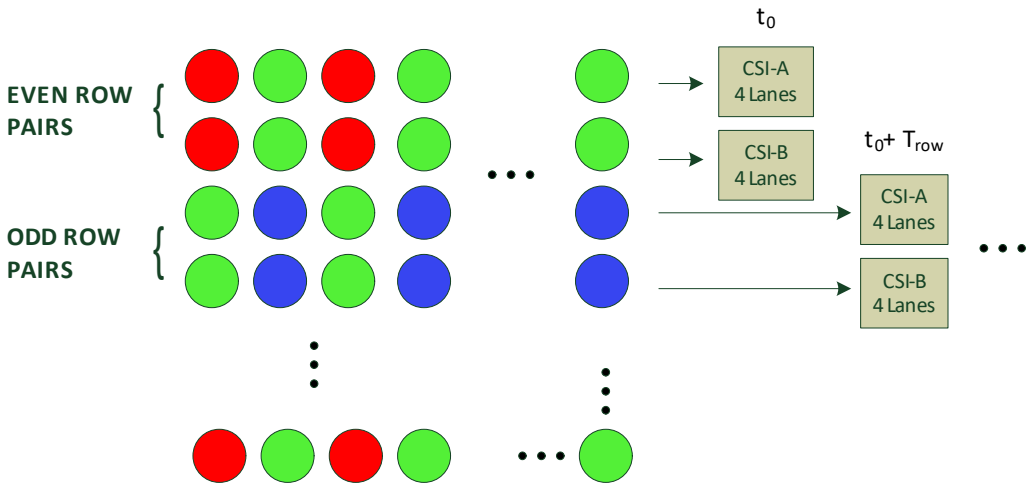


Interleaved Capture Model

The interleaved capture model supports image data that is presented as a standard RGGB Bayer pattern with either the even or odd rows available through one of the CSI ports and the odd or even rows through another of the CSI ports. The VI3 unit provides the RAW10 image data to system memory.

Utilizing two of the CSI 4-lane MIPI interfaces allows an effective still capture data rate to be doubled since all 8 MIPI lanes are active simultaneously. In this mode, the data is presented to the CSI interface in a unique fashion with even row pairs, both consisting of Red/Green pixels presented to one of the three CSI instance MIPI 4-lane interfaces, and one of the two remaining CSI instance, MIPI 4-lane interfaces, followed by the odd row pairs consisting of Green/Blue pixels.To provide a standard RGGB Bayer pattern in memory, an interleaving mechanism is provided as the data is written to memory. The following figure illustrates the even and odd row pair's presentation to the CSI with the first and second 4-lane interfaces referred to as CSI-A and CSI-B, respectively.

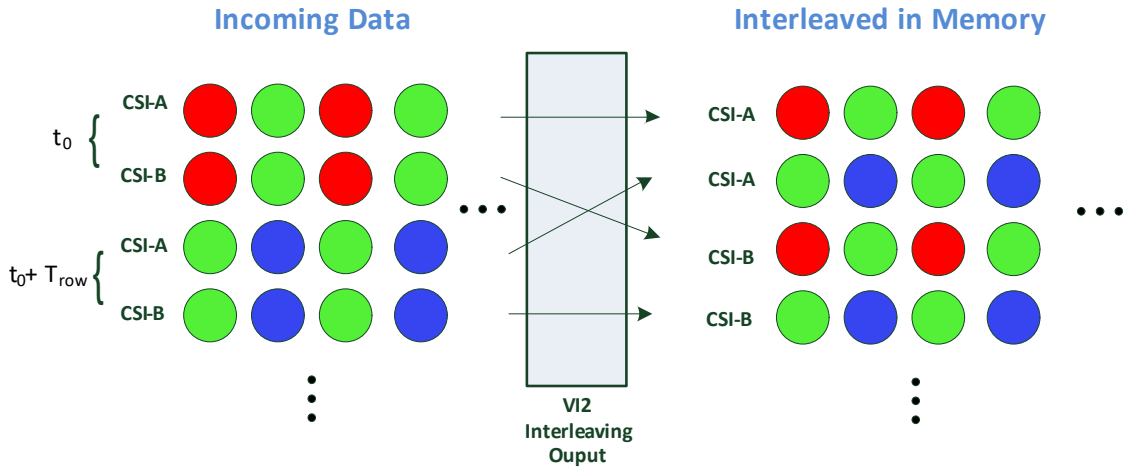
Figure 135: Interleaved Capture Data Presentation



Dual-Channel Interleaving

The interleaved still image is delivered in even and odd row pairs resulting in two Red/Green or two Blue/Green rows being presented to memory at the same time. The preparation of the data for processing involves interleaving the rows so that a Bayer pattern organization of the row data is available in system memory. To accomplish this, the VI3 unit provides a special interleaved mode in which the line counters for the two streams are incremented in a predetermined sequence. The following figure illustrates the interleaving of the incoming data stream in system memory.

Figure 136: Dual-Channel Interleaving During Write to Memory



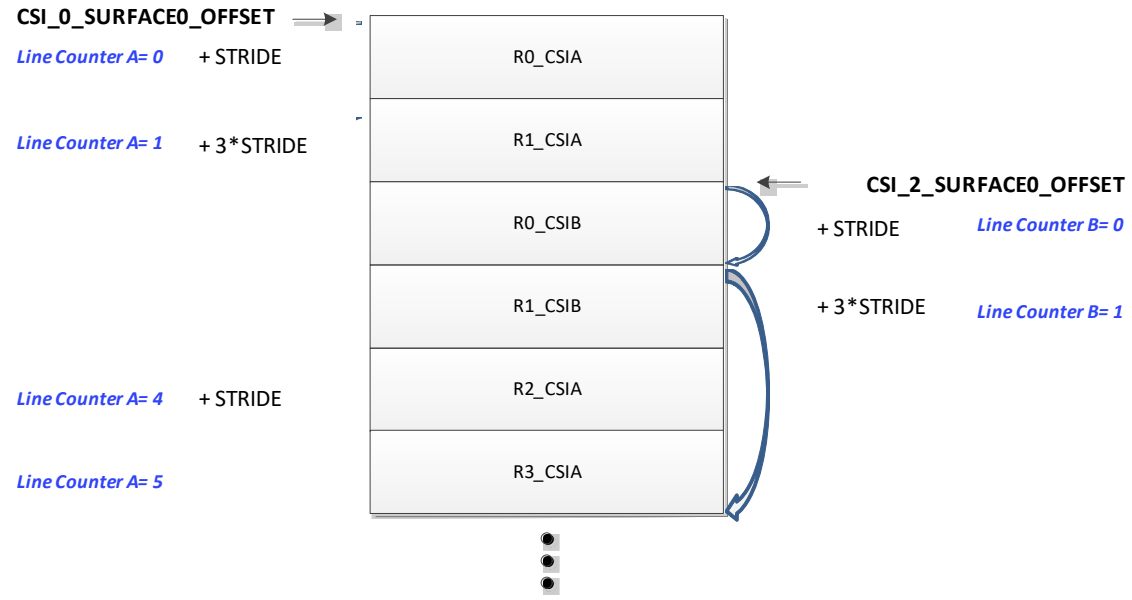
The interleaved mode is enabled by setting the INTERLEAVING_MODE bit in the appropriate VI_CSI_[0..5]_IMAGE_DEF registers. The mode setting changes the normal operation of the line counters used to calculate the memory address after a line is written to increment in a line pair skipping fashion as opposed to the normal mode of single increments.

Figure 137: Line Counter Sequence (Normal and Interleaved Modes)

Captured Row delivered by CSIA or CSIB	NORMAL MODE LINE COUNTER SEQUENCE	INTERLEAVED MODE LINE COUNTER SEQUENCE
Row 0	0	0
Row 1	1	1
Row 2	2	4
Row 3	3	5
Row 4	4	8
Row 5	5	9
.	.	.
.	.	.
.	.	.
Row (Height-1)	HEIGHT-1	HEIGHT-1

The base address and stride for each of the channels are programmed prior to initiating the image capture. Once the image capture is initiated, the line counters for the two channels will sequence in a row pair fashion as illustrated in the following figure.

Figure 138: Dual Channel Interleaving in Memory for T_R16_



The RAW10 format is used for the interleaving capture and may be stored in either the T_R16_I or the T_L10_L10_L10_X2 formats. In addition to the INTERLEAVING_MODE selection, six registers must be programmed to manage the writing of the data to memory.

Table 183: Dual-Channel Interleaving Mode Address and Stride Programming

Register	Data Format	Comment
VI_CSI_0_SURFACE0_OFFSET_LSB VI_CSI_0_SURFACE0_OFFSET_MSB	T_L16_I or T_L10_L10_L10_X2	Base Address for the first even row pair still image in system memory. Only atom aligned (64 Bytes) address is supported
VI_CSI_2_SURFACE0_OFFSET_LSB VI_CSI_2_SURFACE0_OFFSET_MSB	T_L16_I or T_L10_L10_L10_X2	Base Address for the odd even row pair still image in system memory Only atom aligned (64 Bytes) address is supported. VI_CSI_0_SURFACE0_OFFSET_LSB + 2 * STRIDE.
VI_CSI_0_SURFACE0_STRIDE0 VI_CSI_2_SURFACE0_STRIDE0	T_L16_I	STRIDE = 2 * IMAGE_WIDTH Only atom aligned (64 Bytes) strides are supported
	T_L10_L10_L10_X2	STRIDE = (4/3) * IMAGE_WIDTH Only atom aligned (64 Bytes) strides are supported

31.2.9.5 Multi-Camera Still and Video Capture

The multi-camera capture model is defined for automotive applications in which there are up to six camera streams that are provided to system memory.

31.2.10 TrustZone Support

The VI3 unit allows management of several CSI registers associated with memory transactions and resets as either freely programmable in the system or only programmable by secure world functions. The registers supporting the secure programming feature include the reset and base, stride, and image definition registers for CSI[0..5]. These TrustZone® support registers are listed below.

- VI_CSI_[0,5]_ SW_RESET
- VI_CSI_[0,5]_ IMAGE_DEF
- VI_CSI_[0,5]_SURFACE0_OFFSET_MSB
- VI_CSI_[0,5]_SURFACE0_OFFSET_LSB
- VI_CSI_[0,5]_SURFACE1_OFFSET_MSB
- VI_CSI_[0,5]_SURFACE1_OFFSET_LSB
- VI_CSI_[0,5]_SURFACE2_OFFSET_MSB
- VI_CSI_[0,5]_SURFACE2_OFFSET_LSB
- VI_CSI_[0,5]_SURFACE0_BF_OFFSET_MSB
- VI_CSI_[0,5]_SURFACE0_BF_OFFSET_LSB
- VI_CSI_[0,5]_SURFACE1_BF_OFFSET_MSB
- VI_CSI_[0,5]_SURFACE1_BF_OFFSET_LSB
- VI_CSI_[0,5]_SURFACE2_BF_OFFSET_MSB
- VI_CSI_[0,5]_SURFACE3_BF_OFFSET_LSB
- VI_CSI_[0,5]_SURFACE0_STRIDE
- VI_CSI_[0,5]_SURFACE1_STRIDE
- VI_CSI_[0,5]_SURFACE2_STRIDE
- VI_CSI_[0,5]_SURFACE_HEIGHT0

Programming control of the secure CSI registers is based on the programmed output security PMC's APBDEV_PMC_STICKY_BITS_0 field: CSI_SECURE_ENABLE and the TrustZone secure mode being active or inactive. Once the CPU is context switched from the normal operation “non-secure world” to the “secure world” (indicated to the rest of the system by the TrustZone_NS (non-secure bit) being deasserted, for example, there is a Host1x trustzone_nonsecure signal), the VI allows for writes to the CSI registers when the sticky bit register is asserted. If the sticky bit is asserted, but the _NS signal is asserted (that is, the CPU is operating in the non-secure world), then the VI’s CSI registers are not writable. If the sticky bit is not asserted at all, then the VI’s CSI registers can be written to normally.

31.3 Data Formatting

The following subsections describe the CSI pixel formats, ISP2 pixel formatting, and the various formats and packing options for delivering data to system memory.

31.3.1 CSI Input Data Formats

The CSI provides data to the VI3 unit in one of several possible formats. The following table describes the different formats presented to the VI3 unit through the CSI interfaces.

Table 184: CSI to VI3 Pixel Formats

Format Name		23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
6-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
7-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D6	D5	D4	D3	D2	D1	D0
8-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
10-bit raw or		0	0	0	0	0	0	0	0	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
12-bit raw		0	0	0	0	0	0	0	0	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
14-bit raw		0	0	0	0	0	0	0	0	0	0	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
8-bit arbitrary/ embedded		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
16-bit RGB (RGB565)		B4	B3	B2	B1	B0	0	0	0	G5	G4	G3	G2	G1	G0	0	0	R4	R3	R2	R1	R0	0	0	0
15-bit RGB (RGB555)		B4	B3	B2	B1	B0	0	0	0	G4	G3	G2	G1	G0	0	0	0	R4	R3	R2	R1	R0	0	0	0
24-bit RGB (RGB888)		B7	B6	B5	B4	B3	B2	B1	B0	G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0
12-bit RGB (RGB444)		B3	B2	B1	B0	0	0	0	0	G3	G2	G1	G0	0	0	0	0	R3	R2	R1	R0	0	0	0	0
18-bit RGB (RGB666)		B5	B4	B3	B2	B1	B0	0	0	G5	G4	G3	G2	G1	G0	0	0	R5	R4	R3	R2	R1	R0	0	0
YUV422 8 -bit	1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 8-bit (legacy)	Odd Line 1st CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Odd Line 2nd CLK	0	0	0	0	0	0	0	0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	0	0	0	0	0	0	0	0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
YUV420 8-bit (CSPS)	Odd Line	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
	Even Line 1st CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
	Even Line 2nd CLK	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10

The formats include various flavors of RAW, RGB, and YCbCr in addition to 8-bit arbitrary or embedded data. The bits marked ‘0’ are actually don’t-care for VI3 and are ignored by the VI3 unit and the odd line in the above table refers to the first line from the CSI unit.

31.3.2 Raw Pixel Data Formatting

The Raw pixel data may be delivered to the ISP, system memory, or both the ISP and system memory simultaneously. In order to interface to the ISP efficiently, pixels are prepared for the ISP pipeline by converting to the unsigned offset binary representation of the form:

I.F

Where:

I is the number of integer bits.

F is the number of fractional bits.

The zero offset shall have a value of $2^{(F + I - 2)}$. For the ISP, it is suggested that the values of I and F should be:

I = 1, F = 13, offset = 4096_{10}

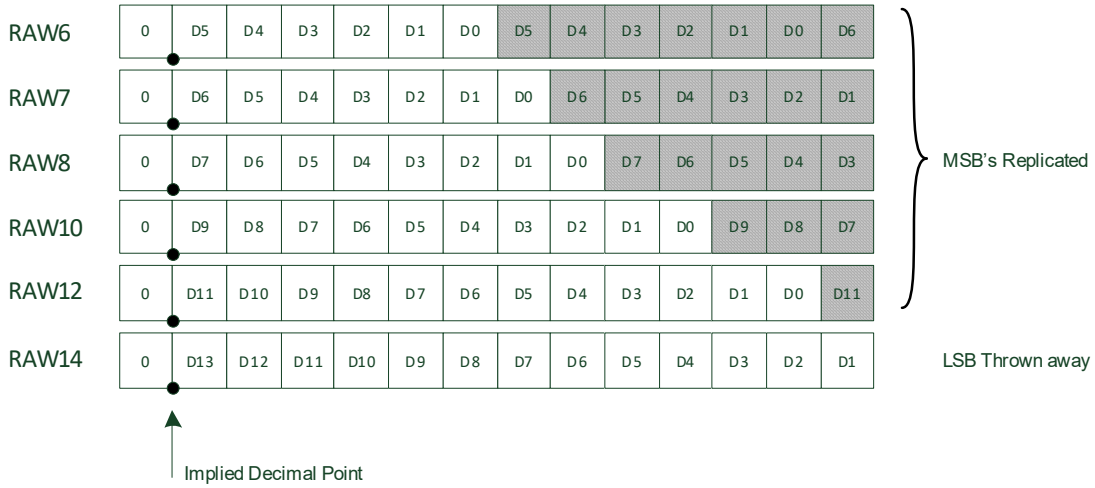
The formatting of the captured RAW data to the 14-bit I.F format requires several steps to be performed. Data may also be stored to memory either simultaneous with the ISP presentation or alone. When storing data to memory, either the formatted data or the original data may be stored to memory. Since the VI3 RAW data streams are all targeted for processing by the ISP, the storing of data which has formatting provided within the VI3 pipeline is the default behavior. Optionally the formatting may be bypassed by setting the BYPASS_PXL_TRANSFORM in the VI_CSI_[0..5]_IMAGE_DEF register.

The formatting steps for preparation of RAW pixel data for the ISP pipeline and/or memory are described in the following sections.

31.3.2.1 Raw Pixel Formatting to VI Pipeline

The Raw data received from the CSI unit is formatted as a 14-bit unsigned operand, U1.13. The data is scaled to provide align the MSB with the implied decimal point. When converting from a source data size that is smaller than the destination data size, the MSBs of the source are replicated to fill the missing LSBs in the destination. The bit replication is shown in the following figure.

Figure 139: VI3 Pipeline Formatting for U1.13 Raw Pixel Data

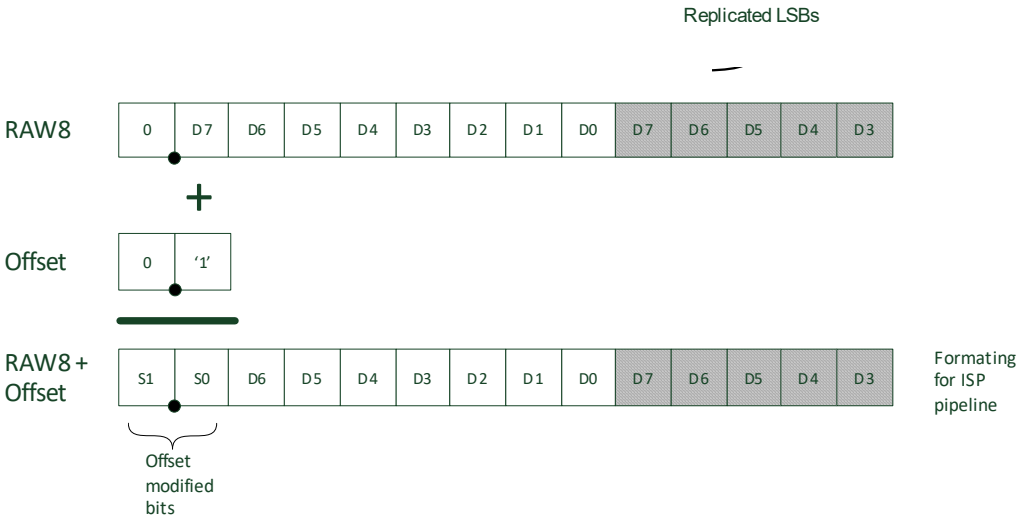


Since the RAW14 data would be wider than the ISP pipe when converting to the unsigned U1.13 format, the LSB is truncated.

31.3.2.2 RAW Pixel Formatting to ISP Pipeline

If the destination format is the pixel bus format, the data must be aligned with the implied decimal point position and the offset must be added. For example the formatting of RAW8 pixel data for the ISP pipeline is illustrated in the following figure.

Figure 140: RAW8 to ISP Pipeline Formatting Example



31.3.2.3 Raw Pixel Formatting to Memory

The raw pixel data may be presented to system memory in three possible formats specific to the input bit precision. The TL_8 formatting operation provides source to destination bit precision increase and decrease. All of the TL_8 formats are packed four per 32-bit word when stored to memory. The T_R16_I employs MSB replication based off of the RAW data previously formatted for the VI pipeline and a new format is introduced for packing three 10-bit raw pixels into a 32-bit word. The different options are summarized in the following table.

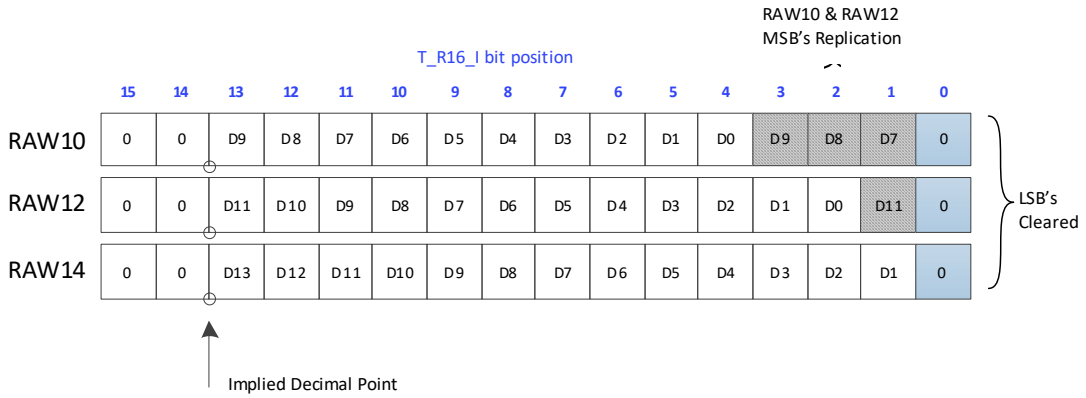
Table 185: Raw Supported Memory Formats

CSI Format Name	Pixel Format in Memory	bpp	Notes
RAW6	T_L8	8	
RAW7	T_L8	8	
RAW8	T_L8	8	
RAW10	T_L8	8	Drop 2 LSBs
	T_R16_I	16	
	T_L10__L10__L10_X2	32	3 RAW10 Pixels Packed
RAW12	T_L8	8	Drop 4 LSBs
	T_R16_I	16	
RAW14	T_L8	8	Drop 6 LSBs
	T_R16_I	16	

T_R16_I Formatting for RAW10 RAW12 and RAW14

The formatting of the raw data from the VI3 pipeline for bit precisions above eight involves an additional MSB bit replication as illustrated in the following figure.

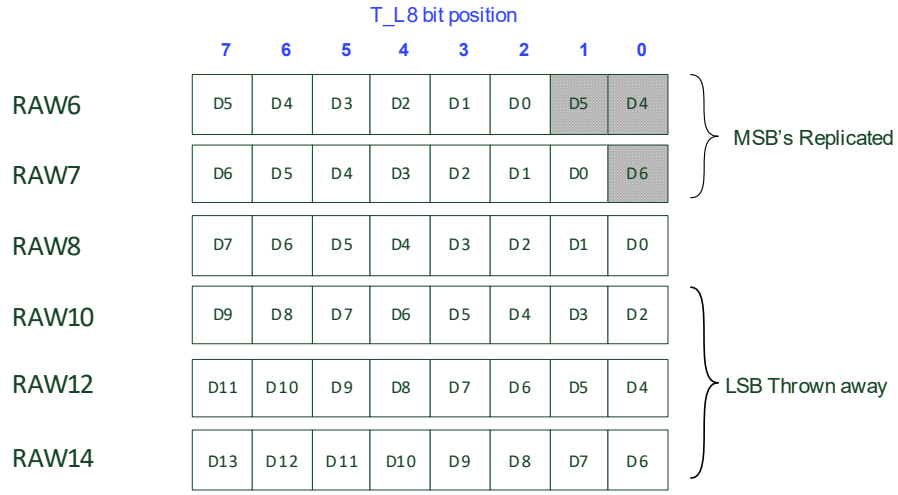
Figure 141: RAW10, RAW12, and RAW14 Formatting to Memory



T_L8_F Formatting for all Raw Pixel Precisions

The formatting for the 8-bit data is a straightforward mapping with either truncation or MSB replication being employed. The formatting for 8-bit data is illustrated in the following figure.

Figure 142: Raw Formatting to T_L8 Memory Format



T_L10__L10__L10_X2 Formatting for RAW10

The T_L10_L10_L10_X2 packed format is introduced to help reduce the burden on the system memory resources during high bandwidth camera user models. In this format, three RAW10 pixels are packed per 32-bit word allowing an improvement in utilization to be realized. The memory efficiency is 93.75% compared to the T_R16_I format with 62.5%.

31.3.3 RGB Pixel Data Formatting

The VI3 unit receives RGB data from the CSI in one of several possible formats and provides for formatting and presentation to system memory. The RGB formats are represented through variation in the number of bits used to represent the red, green, and blue channels. The RGB memory formats and options for source to destination formatting are described in the following table.

Table 186: RGB Supported Memory Formats

CSI Format Name	Pixel Formats in Memory	bpp	Notes
RGB565	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_B5G6R5; T_R5G6B5	16	

Table 186: RGB Supported Memory Formats

CSI Format Name	Pixel Formats in Memory	bpp	Notes
RGB555	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_B5G6R5; T_R5G6B5	16	
RGB888	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_A4B4G4R4; T_A4R4G4B4; T_B4G4R4A4; T_R4G4B4A4	16	
	T_B5G6R5; T_R5G6B5	16	
	T_A8B8G8R8; T_A8R8G8B8; T_B8G8R8A8; T_R8G8B8A8	32	
	T_A2B10G10R10; T_A2R10G10B10; T_B10G10R10A2; T_R10G10B10A2	32	
RGB444	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1;	16	
	T_A4B4G4R4; T_A4R4G4B4; T_B4G4R4A4; T_R4G4B4A4	16	
	T_B5G6R5; T_R5G6B5	16	
RGB666	T_L8	8	(C1*R+C2*G+C3*B)/16
	T_A1B5G5R5; T_A1R5G5B5; T_B5G5R5A1; T_R5G5B5A1	16	
	T_B5G6R5; T_R5G6B5	16	
	T_A8B8G8R8; T_A8R8G8B8; T_B8G8R8A8; T_R8G8B8A8	32	

Each of the capture RGB formats can be formatted by adjusting the precision of the channels, ordering the channels differently, or extracting a luminance approximation into a packed T_L8 format.

31.3.3.1 RGB Pixel Data Size Conversions

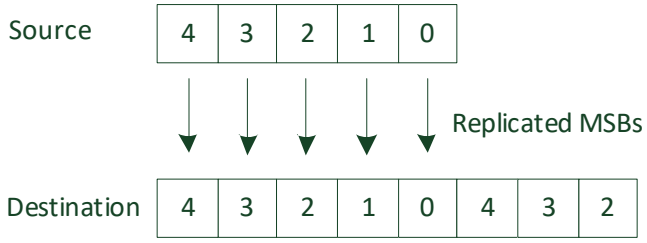
The VI3 unit supports the conversion of the received RGB data into other formats by either increasing or decreasing the precision of the data representing each of the channels. The following table describes the possible conversions.

Table 187: RGB Data Precision Adjustments Options

RGB Format Received	RGB Format Conversion (Precision Increase)	RGB Format Conversion (Precision Decrease)
RGB444	RGB555, RGB565	None
RGB555	RGB565	None
RGB565	None	RGB555
RGB666	RGB888	RGB555, RGB565
RGB888	RGB101010	RGB444,RGB555, RGB565, RGB666

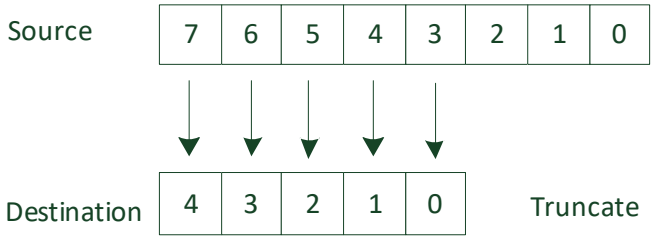
The RGB pixel data is converted to higher bit precision through replication of the MSBs in the source to fill in the missing LSBs in the destination in a similar fashion as the Raw data formatting for the ISP pipe. For example:

Figure 143: 5-bit to 8-bit Precision Increase



When converting from a source data size that is larger than the destination size, the destination is a truncated version of the source data. For example:

Figure 144: 8-bit to 5-bit Precision Decrease



31.3.3.2 RGB to Luma Data Formatting

In addition to providing format conversions, the RGB data may also be stored as luma data by approximating a luminance calculation. This is useful for applications that would otherwise have to compute the luminance channel from the RGB data in a separate step. The luma calculation is performed by multiplying each of the red, green, and blue channel components by a U0.6 coefficient as shown in the following pseudo code.

```
#define SHIFT      5
#define COEFF_BITS 6
    Y = ( (PIXEL.RED_CR * MW_R2Y_COEFF)
        + (PIXEL.GRN_Y * MW_G2Y_COEFF)
        + (PIXEL.BLU_CB * MW_B2Y_COEFF) ) >> (SHIFT + COEFF_BITS);
if (Y < 0)   Y =   0;
if (Y > 255) Y = 255;
```

31.3.3.3 RGB Pixel Formatting to ISP Pipeline

If the destination format is the pixel bus format, the data must be aligned with the implied decimal point position and the offset must be added. For example, the formatting of RGB565 pixel data for the ISP pipeline will involve formatting each of the color components into the 14-bit I.F format.

31.3.4 YUV Pixel Data Formatting

The VI3 unit provides for formatting and packing of YCbCr data received from the CSI and is responsible for its presentation to system memory or to the ISP for scaling or image enhancement. There are two formats captured: YUV422 and YUV420. The formatting provided for YUV capture data is shown in the following table.

Table 188: YUV Supported Memory Formats

CSI Format Name	Pixel Formats in Memory	bpp (per plane)	Notes
YUV422 8-bit	T_L8	8	Y component only
	T_Y8_U8__Y8_V8 [a] ; T_Y8_V8__Y8_U8 [a] ; T_U8_Y8__V8_Y8 [a] ; T_V8_Y8__U8_Y8 [a]	32	
	T_Y8___U8V8_N422	8	Semi-planar; Y+UV
	T_Y8___V8U8_N422	8	Semi-planar; Y+VU
	T_Y8___U8___V8_N422 [b]	8	Planar
YUV420 8-bit legacy	T_L8	8	Y component only
YUV420 8-bit / 8-bit (CSPS)	T_L8	8	Y Component only.

The 8-bit YUV422 format is designated a primary format in the MIPI CSI specification and all others are considered as secondary formats. The supported pixel layout will be Pitch Linear or planar pitch-linear. YUV420 is always stored as T_L8 with even and odd lines stored to memory as they are received from the CSI interface.

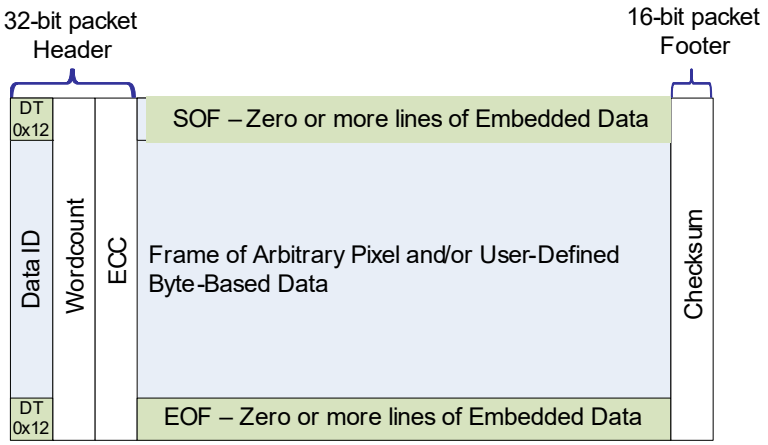
31.3.4.1 YUV Pixel Formatting to ISP Pipeline

The formatting of YUV422 data for presentation to the ISP follows a similar formatting as is done with the raw and RGB formats to convert the data to the unsigned offset binary representation with the exception that an offset of 8192 is used for the chroma channels and an offset of 4096 is used for the luma. The LSB replication follows the same approach as with the RGB and RAW formats. Following the conversion of the luma and chroma to the 14-bit representation for the ISP pipe, the data is then converted from 422 to 444 sampling through interpolation.

31.3.5 Embedded Data Formats

A number of sensor and camera modules utilize embedded data to communicate meta data or calibration data associated with an image capture to the host application. The inclusion of embedded data in the data stream is defined in the MIPI CSI2 specification and has a specific data type of 0x12 assigned. If the embedded data exists, it is identified by the embedded data type at the beginning of a long packet resulting in extra lines added to the beginning or end of a frame. An illustration of embedded and fare data is shown in the following figure.

Figure 145: CSI-2 Incoming Frame Structure with Embedded Data



The CSI and VI3 units work together to manage the capture and handling of the embedded data during a frame capture. Three basic modes of operation associated with the embedded data capture model are supported. The embedded data capture modes are described in the following section.

31.3.5.1 Embedded Data Capture Modes

The CSI[,1,2] can discard embedded data in the CSI[,1,2], to pass embedded data through one pixel parser with the image data being presented to the other pixel parser, or to combine the embedded data with the frame data when presented to a single pixel parser.

Embedded data is discarded by setting CSI[,1,2]_PP[A..B]_EMBEDDED_DATA_OPTION to DISCARD in the CSI[,1,2]_PIXEL_STREAM_[A..B]_CONTROL_0 register. If it is desired to keep the embedded data, this field should be set. Depending on the mode of operation and the data format the embedded data, the frame data, or the combined embedded and frame data may be delivered to system memory or the ISP execution resources. The options for embedded data capture are described in the following table.

Table 189: Embedded Data Capture Modes

EMBEDDED	Pixel Format in Memory	Notes
DISCARD	Either PP will deliver frame data only.	MEM or ISPA/ISPB
Embedded Capture with One PP	Embedded Data and Frame Data using Same PP	MEM only
Embedded Capture with Two PPs	One PP used for streaming Embedded Data.	MEM only
	One PP used for streaming Frame data.	MEM or ISPA/ISPB

If the mode is set to discard, the embedded data is discarded in the CSI instance and only image data is presented to the VI3 unit resulting in the normal data processing with the frame data formatting and presentation to memory and/or the ISP. If the mode is set to capture embedded data using one pixel parser, the embedded data is streamed with the image data using the same pixel parser. In this mode the embedded data is combined with the frame data and presented to system memory. The third mode uses both pixel parsers with one pixel parser managing the embedded data and the other managing the frame data.

The discard embedded capture is relatively straightforward and only requires the CSI[,1,2]_PP[A..B]_EMBEDDED_DATA_OPTION to be set to discard with all other programming settings the same as during non-embedded frame capture. The other two modes have some complexity associated with the programming and how the data is organized in memory. The following sections describe the single and dual Pixel Parser (PP) modes in more detail.

Embedded Capture using One PP

The embedded capture using one pixel parser will result in embedded data being stored with the frame data in system memory. Since the line width for the 8-bit embedded data may be longer or shorter than the line with for the pixel data, the line width short/long errors are ignored. In addition, the CSI[,1,2] will signal to the VI3 unit that an incoming line is embedded by setting the CSI[,1,2][A,B]2VI_FRAME_ERROR during the start of a line. In this fashion, the VI3 will know that the line is embedded and provide for managing the capture and providing the specialized formatting depending on the VI3 memory format setting. The register programming is similar to non-embedded cases except that:

1. VI_CSI_[0..5]_CSI_IMAGE_SIZE register HEIGHT should be programmed with the SOF Embedded Lines+EOF Embedded lines +Normal Lines
2. The CSI_PP[A..B]_EMBEDDED_DATA_OPTION should be set to EMBEDDED in the CSI[,1,2]_PIXEL_STREAM_[A..B]_CONTROL_0 register

In the single PP embedded capture mode, the only destination supported is to MEM since the ISP does not support embedded data with the pixel stream. There are several additional constraints on the supported data formats when using this embedded capture option:

1. RGB Embedded Data capture is not supported.
2. Several Raw variations with various output options are not supported.
3. YUV embedded capture will combine the embedded data with the Luma plane, and insert zeros into the Chroma planes for some formatting options.

Embedded Capture using Two PPs

The embedded capture using two pixel parsers results in one of the pixel parsers delivering only embedded data to the VI3 unit and the other delivering only image data. In order to accomplish this, one PP would be programed to parse pixel data and discard embedded data, and the other PP would be programmed to parse only embedded data. The embedded data would then be sent to memory for the VI3 and the pixel data could either be presented to system memory or the ISP.

For example, if an embedded image stream is being captured from CILA, the pixel path could be set up to be CILA → PPA →VI and the embedded data path could be set up to be CILA→PPB→VI→ MEM. The PPA stream programming is similar to normal cases and the stream B programming would be as follows:

1. Route CILA to PPB by setting the CSI_PPB_STREAM_SOURCE to CSI_A
2. Program VI_CSI_1_CSI_IMAGE_SIZE register HEIGHT to be the sum of SOF Embedded Lines+EOF Embedded lines.
3. Program VI_CSI_1_CSI_IMAGE_SIZE register WIDTH to be based on the pixel data line width/data format.
4. Set the DATA_TYPE to EMBED in the VI_CSI_1_CSI_IMAGE_DT register.
5. Set the CSI_PP[A..B]_EMBEDDED_DATA_OPTION to EMBEDDED in the CSI_PIXEL_STREAM_[A..B]_CONTROL_0 register

The programming of the image width in step three is sometimes different than the image width of the normal frame data since the embedded data lines have the same length in bytes as the image data. For example, for a RAW6 64x64 image, there are 48 bytes in each long packet payload resulting in a line width of 48 bytes for the embedded data. Embedded data is always stored as T_L8 in this mode.

31.3.5.2 Frame Data Type with Embedded Data Limitations

The embedded data when using the two pixel parser option is straightforward as long as the frame height and width are programmed appropriate to the data type and the number of embedded lines. The single PP mode of embedded capture is more complex and software will be required to parse the embedded data from the stored embedded data plus image data frame.

The Raw embedded data capture options supported with a single pixel parsers are summarized in the following table.

Table 190: Raw and DPCM Embedded Data Formatting

CSI Format	VI Pixel Format in Memory	1 PP Embed Mode Support	Embedded Data Layout in 32-bit DWORD
RAW6	T_L8	No	NA
RAW7	T_L8	No	NA
RAW8	T_L8	Yes	[emdat3, emdat2, emdat1, emdat0]
RAW10 or DPCM 10-8-10	T_L8	No	NA
	T_R16_I	Yes	{4'h0,embed_data1,4'hX,4'h0,embed_data0,4'hX}
	T_X2Lc10Lb10La10	Yes	{2'b00,embed_data2,2'b00,embed_data1,2'b00,embed_data0}
RAW12 or DPCM 12-8-12	T_L8	No	NA
	T_R16_I	Yes	{6'h0,embed_data1,2'hX,6'h0,embed_data0,2'hX}

Table 190: Raw and DPCM Embedded Data Formatting

CSI Format	VI Pixel Format in Memory	1 PP Embed Mode Support	Embedded Data Layout in 32-bit DWORD
RAW14 or DPCM 14-8-14	T_L8	No	NA
	T_R16_I	No	NA

The YUV420 format is straightforward since only the Luma plane capture is supported. In this capture mode, embedded data is formatted to look like extra rows of equal length as the Luma data since both are packed T_L8. The YUV422 formatting is significantly more complex and some care is necessary to properly parse the data. The YUV embedded data formatting using a single PP is summarized in the following table.

Table 191: YUV Embedded Data Formatting

CSI Format	VI Pixel Format in Memory	1 PP Embed Mode Support	Embedded Data Layout in 32-bit DWORD
YUV420	T_L8	Yes	[emdat3, emdat2, emdat1, emdat0]
YUV422_8	T_L8	Yes	[emdat3, emdat2, emdat1, emdat0]
	T_Y8_U8__Y8_V8	Yes	{8'h0, embed_data1, 8'h0, embed_data0}
	T_U8_Y8__V8_Y8	Yes	{embed_data1, 8'h0, embed_data0, 8'h0}
	T_Y8_V8__Y8_U8	Yes	{8'h0, embed_data1, 8'h0, embed_data0}
	T_V8_Y8__U8_Y8	Yes	{embed_data1, 8'h0, embed_data0, 8'h0}
	T_Y8__U8V8_N422	Yes	{embed_data3,embed_data2,embed_data1,embed_data0} in Y buffer, The Embedded lines in U/V buffer will be zeroes
	T_Y8__U8V8_N422	Yes	{embed_data3,embed_data2,embed_data1,embed_data0} in Y buffer, The Embedded lines in U/V buffer will be zeroes
	T_Y8__U8__V8_N422	Yes	{embed_data3,embed_data2,embed_data1,embed_data0} in Y buffer, The Embedded lines in U/V buffer will be zeroes

31.3.6 Pixel Formats Memory Layout

31.3.6.1 Non-Planar Formats

The bit arrangements of the color components within a pixel for all the supported non-planar color formats are shown in the table below.

Table 192: Pixel Bit Assignments for Non-Planar Formats

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_L8																									7	6	5	4	3	2	1	0
T_R16_I																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_R5G6B5																	4	3	2	1	0	5	4	3	2	1	0	4	3	2	1	0
T_B5G6R5																	4	3	2	1	0	5	4	3	2	1	0	4	3	2	1	0
T_R5G5B5A1																	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0	A0
T_B5G5R5A1																	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0	A0
T_A1R5G5B5																	A0	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0
T_A1B5G5R5																	A0	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0
T_R4G4B4A4																	3	2	1	0	3	2	1	0	3	2	1	0	A3	A2	A1	A0
T_B4G4R4A4																	3	2	1	0	3	2	1	0	3	2	1	0	A3	A2	A1	A0
T_A4R4G4B4																	A3	A2	A1	A0	3	2	1	0	3	2	1	0	3	2	1	0

Table 192: Pixel Bit Assignments for Non-Planar Formats

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_A4B4G4R4																	A3	A2	A1	A0	3	2	1	0	3	2	1	0	3	2	1	0
T_R8G8B8A8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_B8G8R8A8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_A8R8G8B8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
T_A8B8G8R8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
T_R10G10B10A2	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	A1	A0
T_B10G10R10A2	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	A1	A0
T_A2R10G10B10	A1	A0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
T_A2B10G10R10	A1	A0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
T_V8U8Y8A8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	A7	A6	A5	A4	A3	A2	A1	A0
T_A8V8U8Y8	A7	A6	A5	A4	A3	A2	A1	A0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_Y8_U8__Y8_V8	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_Y8_V8__Y8_U8	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
T_U8_Y8__V8_Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0
T_V8_Y8__U8_Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	7	6	5	4	3	2	1	0
T_R16_G16_B16_I	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_L10__L10__L10_X2			9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
T_X2L6L6L6L6L6			5	4	3	2	1	0	5	4	3	2	1	0	5	4	3	2	1	0	5	4	3	2	1	0	5	4	3	2	1	0

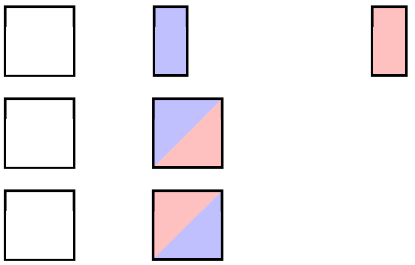
31.3.6.2 Pixel Formats (Planar and Semi-Planar)

Some pixel formats require that the individual color components are separated out into distinct buffers in memory. These formats are defined below:

Table 193: Planar and Semi-Planar Format Definitions

Name	FOURCC	Organization	Sample structure
T_Y8___U8___V8_N422	YV16	Planar	4:2:2
T_Y8___U8V8_N422	NV16	Semi-planar	4:2:2, U followed by V
T_Y8___V8U8_N422	NV61	Semi-planar	4:2:2, V followed by U

Figure 146: Planar and Semi-Planar Buffer Layout Diagram



31.4 VGP (GPIO) Interface

The VI3 unit provides for six programmable GPIOs to control various aspects of the camera system. All may be configured as either I/O or PWM and two may also be configured for I^2C camera control.

Table 194: VGP1-VGP6 Functions

GPIO	Functions		
	A	B	C
VGP1	I/O	PWM	I2C_SCK
VGP2	I/O	PWM	I2C_SDA
VGP3	I/O	PWM	-
VGP4	I/O	PWM	-
VGP5	I/O	PWM	-
VGP6	I/O	PWM	-

The Pulse Width Modulation (PWM) unit generates a series of 128 identical pulses. The unit can use any of the (VGP1-6) pins as an output, provided that the unit is enabled and the pin is configured to carry PWM data. A pulse's length (in clock cycles), duty cycle, and starting polarity are programmable. A single pulse can take between 2 and 32 clock cycles, where a high or low level is no more than 16 cycles long. A series' length is always 128 times the length of a single pulse. Once the unit is enabled, a series can be either generated once, a set number of times, or repeatedly until the unit is disabled.

31.5 Error Handling

The errors can be broadly classified in the categories provided in the following table.

Table 195: VI Error Categories

Error Categories	Description
CSI Error Indication	D-PHY Level Errors
	Packet Level Errors
	Protocol Decoding Errors
Bad Frame Size	Pixel Per Line Error
	Lines Per Frame Error
Overflow Conditions	Back pressure from ISP3.1
	Backpressure from MCCIF
	Backpressure from ISP in ZSL and SZSL Modes
	Backpressure from MCCIF in ZSL and SZSL Modes
Control Flag Error from CSI	Error in control flag sequence

The rest of this section describes details of various errors and their handling by the VI.

31.5.1 SOF Followed by EOF

This is not really an error condition. When the CSI is trying to capture a frame while in single shot mode, if it instead comes across a different frame (not matching the expected parameters of VC/DT/WC), it still sends the SOF indication for the 'bad' frame it dropped and then follows it up with the EOF. This results in the VI receiving an EOF marker without any pixel data or other control flags in between. On receiving this, the VI (depending on the HOSTIF_UPDATE.CAPTURE_GOOD_FRAME configuration) either updates the pending assembly state to active state or overwrites the SOF control flag in the input staging registers.

31.5.2 Missing Control Flags

The missing control flag error occurs when an SOF-SOL-EOL-SOL-EOL-..EOF-SOF sequence occurs. Since the VI does not check for SOL, any error in the sequence concerning SOL is ignored.

31.5.3 Pixels per Line Errors

Pixels Per Line Greater than Image Width

This error is for the case where the first line of the frame was correct and one or more intermediate lines were too long. The CSI will drop any line that does not have the matching WC in the header. This dropping of lines by the CSI might result in a frame-too-short scenario and should be handled as such. The CSI should indicate to the VI when it drops any line via the error flag along with the EOF for that frame. The VI checks for line width and, in case of any error, ignores the rest of the frame.

Pixels Per Line Greater than Image Width

This error is for the case where the first line of the frame was correct and one or more intermediate lines were shorter than expected. This dropping of lines by the CSI might result in a frame-too-short scenario and should be handled as such. The CSI should indicate to VI when it drops any line via the error flag along with the EOF for that frame. The VI checks for line width and, in case of any error, ignores the rest of the frame.

31.5.4 Lines per Frame Errors

Line per Frame Greater than Image Height

In this condition, the input logic should drop the lines beyond the expected frame height and indicate the error to the gatekeeper. On the output side, this error indication is used by output logic to mark the EOF with 'frame too long' error logic to ISP. When sending the frame to memory, the VI should drop the additional lines (beyond the expected height) and not corrupt the memory buffer.

Line per Frame Less than Image Height

In this condition, the input logic does not do any special handling. It may mark the EOF with the error flag (not necessarily required as the read logic can identify the frame too short condition too). The input logic still needs to indicate the error to the gatekeeper module for appropriate handling of next set of assembly registers based on CAPTURE_GOOD_FRAME configuration. On the output side of the line-store buffer, the EOF to ISP is marked with frame-too-short error flag.

31.5.5 CSI Error Indication

In this condition, the error flag is used by the gatekeeper module for appropriate handling (update from assembly state or not as defined by the CAPTURE_GOOD_FRAME configuration). The rest of the frame handling follows normal handling or as defined by other defined error-handling scenarios.

31.5.6 Line Buffer Overflow

In case the backpressure from ISP/MC causes the line buffer to overflow, the incoming pixel data from the CSI is dropped until the EOF. On the read side of line-store buffer, if a premature SOF is detected (since the rest of the frame was dropped at the input), the OP*_DONE syncpt is generated as programmed by the CAPTURE_GOOD_FRAME configuration.

31.6 VI Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

31.6.1 VI_CFG_VI_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	VI_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = VI_CSI_PPA_LINE_START 5 = VI_CSI_PPA_FRAME_START 6 = VI_MWA_REQ_DONE 7 = VI_MWA_ACK_DONE 8 = VI_CSI_PPB_LINE_START 9 = VI_CSI_PPB_FRAME_START 10 = VI_MWB_REQ_DONE 11 = VI_MWB_ACK_DONE 12 = VI_CSI_PPC_LINE_START 13 = VI_CSI_PPC_FRAME_START 14 = VI_MWC_REQ_DONE 15 = VI_MWC_ACK_DONE 16 = VI_CSI_PPD_LINE_START 17 = VI_CSI_PPD_FRAME_START 18 = VI_MWD_REQ_DONE 19 = VI_MWD_ACK_DONE 20 = VI_CSI_PPE_LINE_START 21 = VI_CSI_PPE_FRAME_START 22 = VI_MWE_REQ_DONE 23 = VI_MWE_ACK_DONE 24 = VI_CSI_PPF_LINE_START 25 = VI_CSI_PPF_FRAME_START 26 = VI_MWF_REQ_DONE 27 = VI_MWF_ACK_DONE 28 = VI_ISPA_DONE 29 = VI_ISPB_DONE 30 = VI_VGP0_RCVD 31 = VI_VGP1_RCVD 32 = COND_32
7:0	0x0	VI_INDX: syncpt index value

31.6.2 VI_CFG_VI_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	VI_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	VI_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

31.6.3 VI_CFG_VI_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VI_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

31.6.4 VI_CFG_CTXSW_0

Context Switch Register

Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

The context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS. Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be preloaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module, and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

31.6.5 VI_CFG_INTSTATUS_0

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

31.6.6 VI_PWM_CONTROL_0

VI Pulse Width Modulation Control

PWM signal generation logic can generate up to 128 pulses per line internally and the PWM pulse select registers determine which of the 128 pulses will be output. Any of the 128 internally generated pulse can be independently selected as output if they occur within one line time.

The PWM signal can be output on the VGP6 pin if VGP6 output is enabled and the output select is set to PWM.

The PWM is triggered by the first VSYNC after the PWM_ENABLE bit has been set.

Offset: 0xe | Byte Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xx00xxxxxxxxxxxxxxx0xxx0)

Bit	Reset	Description
31:24	0x0	PWM_COUNTER: 8-bit PWM Counter value used when PWM_MODE is set to COUNTER to determine how many times the PWM will cycle through the 128 cycles before stopping.
21:20	0x0	PWM_MODE: PWM Mode Continuous: After the PWM is turned on, continue through the PWM's 128 cycles repeatedly until the PWM is turned off. Single: After the PWM is turned on, cycle once through the 128 cycles and stop. Counter: After the PWM is turned on, cycle through the 128 cycles PWM_COUNTER number of times then stop. 0 = CONTINUOUS 1 = SINGLE 2 = COUNTER
4	0x0	PWM_DIRECTION: PWM Direction 0= Incrementing 1= Decrementing 0 = INCR 1 = DECR
0	0x0	PWM_ENABLE: PWM Enable 0 = DISABLED 1 = ENABLED

31.6.7 VI_CFG_PWM_HIGH_PULSE_0

PWM High Pulse Period

Offset: 0xf | Byte Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PWM_HIGH_PULSE: PWM High Pulse

31.6.8 VI_CFG_PWM_LOW_PULSE_0

PWM Low Pulse Period

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PWM_LOW_PULSE: PWM Low Pulse

31.6.9 VI_CFG_PWM_SELECT_PULSE_A_0

PWM Pulse Select A

The next 4 registers select which of the internal 128 pulses to output. Each bit in the four registers corresponds to one internal pulse.

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_A: PWM Select bits 31 to 0

31.6.10 VI_CFG_PWM_SELECT_PULSE_B_0

PWM Pulse Select B

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_B: PWM Select bits 63 to 32

31.6.11 VI_CFG_PWM_SELECT_PULSE_C_0

PWM Pulse Select C

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_C: PWM Select bits 95 to 64

31.6.12 VI_CFG_PWM_SELECT_PULSE_D_0

PWM Pulse Select D

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_D: PWM Select bits 127 to 96

31.6.13 VI_CFG_VGP*n*_0

VI VGP*n* Input/Output Config/Data

There are six VI VGP*n* Config registers, where *n* = 1 through 6.

Offset: 0x19 + (*n* – 1) | Byte Offset: 0x64 + (*n* – 1)*0x4 | Read/Write: R/W | Reset: 0x00000X0X (0bxxxxxxxx0xxxxxx00xxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
24	RW	0x0	VGP*n*_INPUT_ENABLE: Reserved
17	RW	0x0	PIN_OUTPUT_SELECT_VGP*n*: Pin Output Select VGP*n* 0 = VG*Pn*_OUTPUT_DATA 1 = 1'b0
16	RW	0x0	VGP*n*_OUTPUT_ENABLE: VGP*n* pin Output Enable. This bit controls the VGP*n* pin output. 0 = DISABLED 1 = ENABLED
8	RO	X	VGP*n*_INPUT_DATA: VGP*n* pin Input Data (effective if VGP*n*_INPUT_ENABLE is ENABLED) 0= VGP*n* input low 1= VGP*n* input high
0	RW	X	VGP*n*_OUTPUT_DATA: VGP*n* pin Output Data (effective if VGP*n*_OUTPUT_ENABLE is ENABLED and VGP*n*_OUTPUT_SELECT is DATA)

31.6.14 VI_CFG_INTERRUPT_MASK_0

Interrupt Mask

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000x)

Bit	Reset	Description
6	0x0	VGP6_INT_MASK: VGP6 pin Interrupt Mask. This bit controls interrupting when a VGP6 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
5	0x0	VGP5_INT_MASK: VGP5 pin Interrupt Mask. This bit controls interrupting when a VGP5 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
4	0x0	VGP4_INT_MASK: VGP4 pin Interrupt Mask. This bit controls interrupting when a VGP4 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
3	0x0	VGP3_INT_MASK: VGP3 pin Interrupt Mask. This bit controls interrupting when a VGP3 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
2	0x0	VGP2_INT_MASK: VGP2 pin Interrupt Mask. This bit controls interrupting when a VGP2 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
1	0x0	VGP1_INT_MASK: VGP1 pin Interrupt Mask. This bit controls interrupting when a VGP1 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED

31.6.15 VI_CFG_INTERRUPT_TYPE_SELECT_0

Interrupt Type Select

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000000x)

Bit	Reset	Description
6	0x0	VGP6_INT_TYPE: VGP6 pin Interrupt Type. This bit controls interrupt VGP6 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
5	0x0	VGP5_INT_TYPE: VGP5 pin Interrupt Type. This bit controls interrupt VGP5 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
4	0x0	VGP4_INT_TYPE: VGP4 pin Interrupt Type. This bit controls interrupt VGP4 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
3	0x0	VGP3_INT_TYPE: VGP3 pin Interrupt Type. This bit controls interrupt VGP3 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
2	0x0	VGP2_INT_TYPE: VGP2 pin Interrupt Type. This bit controls interrupt VGP2 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL
1	0x0	VGP1_INT_TYPE: VGP1 pin Interrupt Type. This bit controls interrupt VGP1 0= Edge type 1= Level type 0 = EDGE 1 = LEVEL

31.6.16 VI_CFG_INTERRUPT_POLARITY_SELECT_0

Interrupt Polarity Select

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000000x)

Bit	Reset	Description
6	0x0	VGP6_INT_POLARITY: VGP6 pin Interrupt Type. This bit controls interrupt VGP6. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH

Bit	Reset	Description
5	0x0	VGP5_INT_POLARITY: VGP5 pin Interrupt Type. This bit controls interrupt VGP5. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
4	0x0	VGP4_INT_POLARITY: VGP4 pin Interrupt Type. This bit controls interrupt VGP4. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
3	0x0	VGP3_INT_POLARITY: VGP3 pin Interrupt Type. This bit controls interrupt VGP3. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
2	0x0	VGP2_INT_POLARITY: VGP2 pin Interrupt Type. This bit controls interrupt VGP2. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
1	0x0	VGP1_INT_POLARITY: VGP1 pin Interrupt Type. This bit controls interrupt VGP1. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH

31.6.17 VI_CFG_INTERRUPT_STATUS_0

Interrupt Enable

This register returns interrupt status when read. When this register is written, the interrupt status corresponding to the bits written with 1 is reset. Interrupt status corresponding to the bits written with 0 will be left unchanged.

Note that interrupt status bits can be set even when their corresponding interrupt enable bits, in the VI_CFG_INTERRUPT_MASK_0 register, are cleared. When these bits are set and their corresponding interrupt enable bits are set, an interrupt is generated. The interrupt can be cleared or left unchanged by writing 1 or 0, respectively, to the corresponding bits in this register.

Clearing the interrupt status bits does not affect the interrupt enable bits.

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	VGP6_INT_STATUS: VGP6 pin Interrupt Status. This bit controls interrupt when a VGP6 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
5	X	VGP5_INT_STATUS: VGP5 pin Interrupt Status. This bit controls interrupt when a VGP5 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
4	X	VGP4_INT_STATUS: VGP4 pin Interrupt Status. This bit controls interrupt when a VGP4 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR

Bit	Reset	Description
3	X	VGP3_INT_STATUS: VGP3 pin Interrupt Status. This bit controls interrupt when a VGP3 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
2	X	VGP2_INT_STATUS: VGP2 pin Interrupt Status. This bit controls interrupt when a VGP2 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
1	X	VGP1_INT_STATUS: VGP1 pin Interrupt Status. This bit controls interrupt when a VGP1 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR

31.6.18 VI_CFG_VGP_SYNCPT_CONFIG_0

Offset: 0x2b | Byte Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	SYNCPT_VGPY_SELECT: Selects the VGP (1-6) for SYNCPT condition VGP_1_RECD
2:0	0x0	SYNCPT_VGPX_SELECT: Selects the VGP (1-6) for SYNCPT condition VGP_0_RECD

31.6.19 VI_CFG_VI_SW_RESET_0

Offset: 0x2d | Byte Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MCCIF_RESET: Resets the MCCIF interface

31.6.20 VI_CFG_CG_CTRL_0

Offset: 0x2e | Byte Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CG_2ND_LEVEL_EN: Second-level clock gating control 0: Disables any second-level clock gating (MCCIF clocks should be controlled from CAR registers) 1: Enables second-level clock gating 0 = DISABLE 1 = ENABLE

31.6.21 VI_CFG_VI_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register (where applicable) and Clock Gating Control

Note: *The FIFO timing aspects of this register are not supported but are retained for software compatibility.*

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to the legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC.

A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00000xxxxxxxxxxxxxx00)

Bit	Reset	Description
20	LEGACY	VI_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	VI_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	VI_CCLK_OVERRIDE
17	0x0	VI_RCLK_OVERRIDE
16	0x0	VI_WCLK_OVERRIDE
1	DISABLE	VI_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	VI_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

31.6.22 VI_CFG_TIMEOUT_WCOAL_VI_0

Write Coalescing Time-Out Register

Note: *Write coalescing is not supported by the MCCIF clients. Registers are retained for backwards compatibility but are not used by the hardware.*

Offset: 0x38 | Byte Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	VIWSB_WCOAL_TMVAL

31.6.23 VI_CFG_DVFS_0

Dynamic Voltage Frequency Shift Register

Offset: 0x3a | Byte Offset: 0xe8 | Read/Write: R/W | Reset: 0x404001ff (0bx1000000x1000000xxxxxxx111111111)

Bit	Reset	Description
30:24	0x40	SENSORB_LB_THRESHOLD: This field defines the DVFS threshold for the VI Line buffer for Sensor B, which is compared against the buffer utilization in order to assert the ready_for_latency event signal to the MC. The final ready_for_latency_signal is ANDed with the one generated from the MCCIF_THRESHOLD comparison.
22:16	0x40	SENSORA_LB_THRESHOLD: This field defines the DVFS threshold for the VI Line buffer for Sensor A, which is compared against the buffer utilization in order to assert the ready_for_latency event signal to the MC. The final ready_for_latency_signal is ANDed with the one generated from the MCCIF_THRESHOLD comparison.
8:0	0xff	MCCIF_THRESHOLD: This field defines the DVFS threshold for the MCCIF FIFO, which is compared against the write request buffer utilization in order to assert the ready_for_latency_event signal to the MC.

31.6.24 VI_CFG_DVFS_1_0

Dynamic Voltage Frequency Shift Register

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0x40404040 (0bx1000000x1000000x1000000x1000000)

Bit	Reset	Description
30:24	0x40	SENSORF_LB_THRESHOLD: This register defines the DVFS threshold for VI Line buffer for Sensor B, which is compared against the buffer utilization in order to assert ready_for_latency event signal to the Memory Controller. The final ready_for_latency_signal is AND'd with the one generated from MCCIF_THRESHOLD comparison.
22:16	0x40	SENSORE_LB_THRESHOLD: This register defines the DVFS threshold for VI Line buffer for Sensor B, which is compared against the buffer utilization in order to assert ready_for_latency event signal to the Memory Controller. The final ready_for_latency_signal is AND'd with the one generated from MCCIF_THRESHOLD comparison.
14:8	0x40	SENSORD_LB_THRESHOLD: This register defines the DVFS threshold for VI Line buffer for Sensor B, which is compared against the buffer utilization in order to assert ready_for_latency event signal to the Memory Controller. The final ready_for_latency_signal is AND'd with the one generated from MCCIF_THRESHOLD comparison.
6:0	0x40	SENSORC_LB_THRESHOLD: This register defines the DVFS threshold for VI Line buffer for Sensor A, which is compared against the buffer utilization in order to assert ready_for_latency event signal to the Memory Controller. The final ready_for_latency_signal is AND'd with the one generated from MCCIF_THRESHOLD comparison.

31.6.25 VI_CFG_DVFSFIFO_CNTRL_0

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8:0	0x0	DVFS_WR_LIMIT_VAL

31.6.26 VI_CFG_RESERVE_0_0

Reserved register

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_0_3
11:8	X	nc_RESERVE_0_2
7:4	X	nc_RESERVE_0_1
3:0	X	nc_RESERVE_0_0

31.6.27 VI_CFG_RESERVE_1_0

Reserved register

Offset: 0x3f | Byte Offset: 0xfc | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_1_3
11:8	X	nc_RESERVE_1_2
7:4	X	nc_RESERVE_1_1
3:0	X	nc_RESERVE_1_0

31.7 VI Input CSI Interface Registers

CSI Channel Registers

31.7.1 VI_CSI_0_SW_RESET_0

Offset: 0x40 | Byte Offset: 0x100 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	ISPINTF_RESET: Reset ISP interface
3	0x0	MCINTF_RESET: Reset Memory Client i/f logic
2	0x0	PF_RESET: Reset Pixel format logic
1	0x0	SENSORCTL_RESET: Reset Sensor control logic
0	0x0	SHADOW_RESET: Reset Shadow copy logic

31.7.2 VI_CSI_0_SINGLE_SHOT_0

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CAPTURE: request update of the previously assembled 'assembly state' to 'active state' and schedule single shot capture for the VI channel. The actual update of operational register is gated by the next EOF Register read back returns pending capture status (at input of VI)

31.7.3 VI_CSI_0_SINGLE_SHOT_STATE_UPDATE_0

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CAPTURE_GOOD_FRAME: 1'b0: update state when the next SOF is received, irrespective of the state of previous frame. 1'b1: update state when the next SOF is received AND the previous frame was the correct frame based on current operational state.

31.7.4 VI_CSI_0_IMAGE_DEF_0

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxx000000000xxxxxx0xxxxx000)

Bit	Reset	Description
24	0x0	BYPASS_PXL_TRANSFORM: Bypass pixel transformation VI unit does convert the pixels into I.F format while interfacing with Memory packer. This bit can be used to bypass this conversion and will useful in writing compressed image format into memory.

Bit	Reset	Description
23:16	0x0	FORMAT: Pixel memory format for the VI channel The following are not supported: T_A2Y10U10V10, T_V10U10Y10A2 T_Y8__U8__V8_N444, T_Y8__U8V8_N444, T_Y8__V8U8_N444 16 = T_L8 32 = T_R16_I 33 = T_B5G6R5 34 = T_R5G6B5 35 = T_A1B5G5R5 36 = T_A1R5G5B5 37 = T_B5G5R5A1 38 = T_R5G5B5A1 39 = T_A4B4G4R4 40 = T_A4R4G4B4 41 = T_B4G4R4A4 42 = T_R4G4B4A4 64 = T_A8B8G8R8 65 = T_A8R8G8B8 66 = T_B8G8R8A8 67 = T_R8G8B8A8 68 = T_A2B10G10R10 69 = T_A2R10G10B10 70 = T_B10G10R10A2 71 = T_R10G10B10A2 193 = T_A8Y8U8V8 194 = T_V8U8Y8A8 197 = T_A2Y10U10V10 198 = T_V10U10Y10A2 200 = T_Y8_U8__Y8_V8 201 = T_Y8_V8__Y8_U8 202 = T_U8_Y8__V8_Y8 203 = T_V8_Y8__U8_Y8 224 = T_Y8__U8__V8_N444 225 = T_Y8__U8V8_N444 226 = T_Y8__V8U8_N444 227 = T_Y8__U8__V8_N422 228 = T_Y8__U8V8_N422 229 = T_Y8__V8U8_N422 230 = T_Y8__U8__V8_N420 231 = T_Y8__U8V8_N420 232 = T_Y8__V8U8_N420 233 = T_X2Lc10Lb10La10 234 = T_A2R6R6R6R6R6
8	0x0	INTERLEAVING_MODE: 1= Enable interleaving mode for writing image into memory
2	0x0	DEST_ISPB: 1= send VI channel data to ISPB
1	0x0	DEST_ISPA: 1= send VI channel data to ISPA
0	0x0	DEST_MEM: 1= send VI channel data to MEM

31.7.5 VI_CSI_0_RGB2Y_CTRL_0

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x00XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:18	X	B2Y_COEFF: Blue coefficient used to convert RGB pixel data to Y for writing to Memory (in U0.6 format)
15:10	X	G2Y_COEFF: Green coefficient used to convert RGB pixel data to Y for writing to Memory (in U0.6 format)
7:2	X	R2Y_COEFF: Red coefficient used to convert RGB pixel data to Y for writing to Memory (in U0.6 format)

31.7.6 VI_CSI_0_MEM_TILING_0

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	TILING_FORMAT: VI channel memory surface tiling format 254 = BLOCK_LINEAR 0 = PITCH_LINEAR

31.7.7 VI_CSI_0_CSI_IMAGE_SIZE_0

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	HEIGHT: Height of VI channel frame in Lines. This Register field is double buffered. Image size is constrained by HEIGHT >= 32 lines and WIDTH*HEIGHT <= 128 MegaPixels
15:0	X	WIDTH: Width of VI channel frame in Pixels. This Register field is double buffered. Image size is constrained by WIDTH > = 32 pixels and WIDTH*HEIGHT < 128 MegaPixels.

31.7.8 VI_CSI_0_CSI_IMAGE_SIZE_WC_0

Offset: 0x47 | Byte Offset: 0x11c | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	WORDCOUNT: VI channel word count for the Image to be captured This parameter specifies the number of bytes per line/packet. This is matched against the Word Count field in packet header if enabled. This count does not include the additional 2 bytes of CRC checksum if data CRC check is enabled. When the input stream comes from a CSI camera port, this parameter must be programmed when CSI_PPA_PAD_SHORT_LINE is set to either PAD0S or PAD1S, no matter whether CSI_PPA_WORD_COUNT_SELECT is set to REGISTER or HEADER. When the input stream comes from the host path or from the VIP path, and the data mode is PAYLOAD_ONLY, this count must be programmed. Given a line width of N pixels, the programming value of this parameters is as follows ---------- data format value ---------- YUV420_8 N bytes YUV420_10 N/4*5 bytes LEG_YUV420_8 N/2*3 bytes YUV422_8 N*2 bytes YUV422_10 N/2*5 bytes RGB888 N*3 bytes RGB666 N/4*9 bytes RGB565 N*2 bytes RGB555 N*2 bytes RGB444 N*2 bytes RAW6 N/4*3 bytes RAW7 N/8*7 bytes RAW8 N bytes RAW10 N/4*5 bytes RAW12 N/2*3 bytes RAW14 N/4*7 bytes This register is used in to set the wordcount in Test Pattern Generation mode. The wordcount size needed to be set in 16-pixel boundary depended on data type selection (RAW10 or RGB888) This Register field is double buffered.

31.7.9 VI_CSI_0_CSI_IMAGE_DT_0

Offset: 0x48 | Byte Offset: 0x120 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
12	0x0	INTERLACED_VIDEO: VI channel interlaced video format enable
9:8	X	VC: VI channel virtual channel ID. This Register field is double buffered.

Bit	Reset	Description
5:0	X	DATA_TYPE: VI channel input data type. This Register field is double buffered. 18 = EMBED 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8 28 = YUV420CSPS_8 29 = YUV420CSPS_10 30 = YUV422_8 31 = YUV422_10 32 = RGB444 33 = RGB555 34 = RGB565 35 = RGB666 36 = RGB888 40 = RAW6 41 = RAW7 42 = RAW8 43 = RAW10 44 = RAW12 45 = RAW14 48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4

31.7.10 VI_CSI_0_SURFACE0_OFFSET_MSB_0

Offset: 0x49 | Byte Offset: 0x124 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_0: Base address MSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.11 VI_CSI_0_SURFACE0_OFFSET_LSB_0

Offset: 0x4a | Byte Offset: 0x128 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_0: Base address LSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.12 VI_CSI_0_SURFACE1_OFFSET_MSB_0

Offset: 0x4b | Byte Offset: 0x12c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_1: Base address MSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.13 VI_CSI_0_SURFACE1_OFFSET_LSB_0

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_1: Base address LSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered.Only atom aligned (64Bytes) address is supported.

31.7.14 VI_CSI_0_SURFACE2_OFFSET_MSB_0

Offset: 0x4d | Byte Offset: 0x134 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_2: Base address MSB for Surface 2 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.15 VI_CSI_0_SURFACE2_OFFSET_LSB_0

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_2: Base address LSB for Surface 2 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.16 VI_CSI_0_SURFACE0_BF_OFFSET_MSB_0

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_0: Base address MSB for Bottom Frame surface 0 in Memory (for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.17 VI_CSI_0_SURFACE0_BF_OFFSET_LSB_0

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_0: Base address LSB for Bottom Frame surface 0 in Memory (for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.18 VI_CSI_0_SURFACE1_BF_OFFSET_MSB_0

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_1: Base address MSB for Bottom Frame surface 1 in Memory (for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.19 VI_CSI_0_SURFACE1_BF_OFFSET_LSB_0

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_1: Base address LSB for Bottom Frame surface 1 in Memory (for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.20 VI_CSI_0_SURFACE2_BF_OFFSET_MSB_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_2: Base address MSB for Bottom Frame surface 2 in Memory (for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.21 VI_CSI_0_SURFACE2_BF_OFFSET_LSB_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_2: Base address LSB for Bottom Frame surface 2 in Memory (for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.22 VI_CSI_0_SURFACE0_STRIDE_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_0: Stride for Surface 0 in Memory. This Register field is double buffered. Only atom aligned (64Bytes) strides are supported.

31.7.23 VI_CSI_0_SURFACE1_STRIDE_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_1: Stride for Surface 1 in Memory. This Register field is double buffered. Only atom aligned (64Bytes) strides are supported.

31.7.24 VI_CSI_0_SURFACE2_STRIDE_0

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_2: Stride for Surface 2 in Memory This Register field is double buffered. Only atom aligned (64Bytes) strides are supported.

31.7.25 VI_CSI_0_SURFACE_HEIGHT0_0

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3:0	0x1	GOBS: Number of GOBs stacked vertically to form a block

31.7.26 VI_CSI_0_ISPINTF_CONFIG_0

Offset: 0x59 | Byte Offset: 0x164 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx011)

Bit	Reset	Description
2:1	0x1	CHROMA_POS: Stream format 0 = MPEG1 1 = MPEG2
0	0x1	DO_YUV_INTERP: DO_YUV_INTERP: 0=Bypass Chroma Interpolator for YUV 1=Run YUV through Chroma Interpolator

31.7.27 VI_CSI_0_ERROR_STATUS_0

Offset: 0x61 | Byte Offset: 0x184 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	WATCHDOG_INT: This event occurs when Watchdog timer expires and the EOF is not received on the CSI2VI interface
4	X	CSI_FRAME_ERROR: Flagged if EOF field from CSI has FRAME_ERROR bit set
3	X	FRAME_HEIGHT_LONG_ERROR: Flagged if frame height is larger than HEIGHT. Write 1 to clear.
2	X	FRAME_HEIGHT_SHORT_ERROR: Flagged if frame height is smaller than HEIGHT. Write 1 to clear.
1	X	LINE_WIDTH_LONG_ERROR: Flagged if frame line width is larger than WIDTH. Write 1 to clear.
0	X	LINE_WIDTH_SHORT_ERROR: Flagged if frame line width is smaller than WIDTH. Write 1 to clear.

31.7.28 VI_CSI_0_ERROR_INT_MASK_0

Offset: 0x62 | Byte Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	WATCHDOG_INT_MASK: Watchdog Timer 0 Interrupt Mask This controls interrupt when WATCHDOG trigger event occurs for the CSI stream 0 = Disabled, 1 = Enabled
4	0x0	CSI_FRAME_ERROR_INT_MASK: CSI error interrupt mask
3	0x0	FRAME_HEIGHT_LONG_INT_MASK: frame height long error interrupt mask
2	0x0	FRAME_HEIGHT_SHORT_INT_MASK: frame height short error interrupt mask
1	0x0	LINE_WIDTH_LONG_INT_MASK: line width long error interrupt mask
0	0x0	LINE_WIDTH_SHORT_INT_MASK: line width short error interrupt mask

31.7.29 VI_CSI_0_WD_CTRL_0

Watch Dog Timer Control Register

Offset: 0x63 | Byte Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	WD_ENABLE: Watch Dog Timer Enable 0 = DISABLE 1 = ENABLE

31.7.30 VI_CSI_0_WD_PERIOD_0

Watch Dog Timer Control Register

Offset: 0x64 | Byte Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	WD_PERIOD: Watch Dog Timer Period in pixel clocks

31.7.31 VI_CSI_1_SW_RESET_0

Offset: 0x80 | Byte Offset: 0x200 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	ISPINTF_RESET: Reset ISP interface
3	0x0	MCINTF_RESET: Reset Memory Client i/f logic
2	0x0	PF_RESET: Reset Pixel format logic
1	0x0	SENSORCTL_RESET: Reset Sensor control logic
0	0x0	SHADOW_RESET: Reset Shadow copy logic

31.7.32 VI_CSI_1_SINGLE_SHOT_0

Offset: 0x81 | Byte Offset: 0x204 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CAPTURE: request update of the previously assembled 'assembly state' to 'active state' and schedule single shot capture for the VI channel. The actual update of operational register is gated by the next EOF Register read back returns pending capture status (at input of VI)

31.7.33 VI_CSI_1_SINGLE_SHOT_STATE_UPDATE_0

Offset: 0x82 | Byte Offset: 0x208 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CAPTURE_GOOD_FRAME: 1'b0: update state when the next SOF is received, irrespective of the state of previous frame. 1'b1: update state when the next SOF is received AND the previous frame was the correct frame based on current operational state.

31.7.34 VI_CSI_1_IMAGE_DEF_0

Offset: 0x83 | Byte Offset: 0x20c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000000 (0bxxxxxxx000000000xxxxxx0xxxxx000)

Bit	Reset	Description
24	0x0	BYPASS_PXL_TRANSFORM: Bypass pixel transformation VI unit does convert the pixels into I.F format while interfacing with Memory packer. This bit can be used to bypass this convertion and will useful in writing compressed image format into memory.

Bit	Reset	Description
23:16	0x0	FORMAT: Pixel memory format for the VI channel The following are not supported: T_A2Y10U10V10, T_V10U10Y10A2 T_Y8__U8__V8_N444, T_Y8__U8V8_N444, T_Y8__V8U8_N444 16 = T_L8 32 = T_R16_I 33 = T_B5G6R5 34 = T_R5G6B5 35 = T_A1B5G5R5 36 = T_A1R5G5B5 37 = T_B5G5R5A1 38 = T_R5G5B5A1 39 = T_A4B4G4R4 40 = T_A4R4G4B4 41 = T_B4G4R4A4 42 = T_R4G4B4A4 64 = T_A8B8G8R8 65 = T_A8R8G8B8 66 = T_B8G8R8A8 67 = T_R8G8B8A8 68 = T_A2B10G10R10 69 = T_A2R10G10B10 70 = T_B10G10R10A2 71 = T_R10G10B10A2 193 = T_A8Y8U8V8 194 = T_V8U8Y8A8 197 = T_A2Y10U10V10 198 = T_V10U10Y10A2 200 = T_Y8_U8__Y8_V8 201 = T_Y8_V8__Y8_U8 202 = T_U8_Y8__V8_Y8 203 = T_V8_Y8__U8_Y8 224 = T_Y8__U8__V8_N444 225 = T_Y8__U8V8_N444 226 = T_Y8__V8U8_N444 227 = T_Y8__U8__V8_N422 228 = T_Y8__U8V8_N422 229 = T_Y8__V8U8_N422 230 = T_Y8__U8__V8_N420 231 = T_Y8__U8V8_N420 232 = T_Y8__V8U8_N420 233 = T_X2Lc10Lb10La10 234 = T_A2R6R6R6R6R6
8	0x0	INTERLEAVING_MODE: 1= Enable interleaving mode for writing image into memory
2	0x0	DEST_ISPB: 1= send VI channel data to ISPB
1	0x0	DEST_ISPA: 1= send VI channel data to ISPA
0	0x0	DEST_MEM: 1= send VI channel data to MEM

31.7.35 VI_CSI_1_RGB2Y_CTRL_0

Offset: 0x84 | Byte Offset: 0x210 | Read/Write: R/W | Reset: 0x00XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:18	X	B2Y_COEFF: Blue coefficient used to convert RGB pixel data to Y for writing to Memory(in U0.6 format)
15:10	X	G2Y_COEFF: Green coefficient used to convert RGB pixel data to Y for writing to Memory(in U0.6 format)
7:2	X	R2Y_COEFF: Red coefficient used to convert RGB pixel data to Y for writing to Memory(in U0.6 format)

31.7.36 VI_CSI_1_MEM_TILING_0

Offset: 0x85 | Byte Offset: 0x214 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	TILING_FORMAT: VI channel memory surface tiling format 254 = BLOCK_LINEAR 0 = PITCH_LINEAR

31.7.37 VI_CSI_1_CSI_IMAGE_SIZE_0

Offset: 0x86 | Byte Offset: 0x218 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	HEIGHT: Height of VI channel frame in Lines. This Register field is double buffered. Image size is constrained by HEIGHT >= 32 lines and WIDTH*HEIGHT <= 128 MegaPixels
15:0	X	WIDTH: Width of VI channel frame in Pixels. This Register field is double buffered. Image size is constrained by WIDTH > = 32 pixels and WIDTH*HEIGHT < 128 MegaPixels.

31.7.38 VI_CSI_1_CSI_IMAGE_SIZE_WC_0

Offset: 0x87 | Byte Offset: 0x21c | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	WORDCOUNT: VI channel word count for the Image to be captured This parameter specifies the number of bytes per line/packet. This is matched against the Word Count field in packet header if enabled. This count does not include the additional 2 bytes of CRC checksum if data CRC check is enabled. When the input stream comes from a CSI camera port, this parameter must be programmed when CSI_PPA_PAD_SHORT_LINE is set to either PAD0S or PAD1S, no matter whether CSI_PPA_WORD_COUNT_SELECT is set to REGISTER or HEADER. When the input stream comes from the host path or from the VIP path, and the data mode is PAYLOAD_ONLY, this count must be programmed. Given a line width of N pixels, the programming value of this parameters is as follows ---------- data format value ---------- YUV420_8 N bytes YUV420_10 N/4*5 bytes LEG_YUV420_8 N/2*3 bytes YUV422_8 N*2 bytes YUV422_10 N/2*5 bytes RGB888 N*3 bytes RGB666 N/4*9 bytes RGB565 N*2 bytes RGB555 N*2 bytes RGB444 N*2 bytes RAW6 N/4*3 bytes RAW7 N/8*7 bytes RAW8 N bytes RAW10 N/4*5 bytes RAW12 N/2*3 bytes RAW14 N/4*7 bytes This register is used in to set the word count in Test Pattern Generation mode. The wordcount size needed to be set in 16-pixel boundary depended on data type selection (RAW10 or RGB888) This Register field is double buffered.

31.7.39 VI_CSI_1_CSI_IMAGE_DT_0

Offset: 0x88 | Byte Offset: 0x220 | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
12	0x0	INTERLACED_VIDEO: VI channel interlaced video format enable
9:8	X	VC: VI channel virtual channel ID. This Register field is double buffered.

Bit	Reset	Description
5:0	X	DATA_TYPE: VI channel input data type. This Register field is double buffered. 18 = EMBED 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8 28 = YUV420CSPS_8 29 = YUV420CSPS_10 30 = YUV422_8 31 = YUV422_10 32 = RGB444 33 = RGB555 34 = RGB565 35 = RGB666 36 = RGB888 40 = RAW6 41 = RAW7 42 = RAW8 43 = RAW10 44 = RAW12 45 = RAW14 48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4

31.7.40 VI_CSI_1_SURFACE0_OFFSET_MSB_0

Offset: 0x89 | Byte Offset: 0x224 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_0: Base address MSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.41 VI_CSI_1_SURFACE0_OFFSET_LSB_0

Offset: 0x8a | Byte Offset: 0x228 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_0: Base address LSB for Surface 0 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.42 VI_CSI_1_SURFACE1_OFFSET_MSB_0

Offset: 0x8b | Byte Offset: 0x22c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_1: Base address MSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.43 VI_CSI_1_SURFACE1_OFFSET_LSB_0

Offset: 0x8c | Byte Offset: 0x230 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_1: Base address LSB for Surface 1 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.44 VI_CSI_1_SURFACE2_OFFSET_MSB_0

Offset: 0x8d | Byte Offset: 0x234 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	OFFSET_MSB_2: Base address MSB for Surface 2 in Memory (or Top Frame for Interlaced Video). This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.45 VI_CSI_1_SURFACE2_OFFSET_LSB_0

Offset: 0x8e | Byte Offset: 0x238 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OFFSET_LSB_2: Base address LSB for Surface 2 in Memory (or Top Frame for Interlaced Video) This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.46 VI_CSI_1_SURFACE0_BF_OFFSET_MSB_0

Offset: 0x8f | Byte Offset: 0x23c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_0: Base address MSB for Bottom Frame surface 0 in Memory (for Interlaced Video) This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.47 VI_CSI_1_SURFACE0_BF_OFFSET_LSB_0

Offset: 0x90 | Byte Offset: 0x240 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_0: Base address LSB for Bottom Frame surface 0 in Memory (for Interlaced Video) This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.48 VI_CSI_1_SURFACE1_BF_OFFSET_MSB_0

Offset: 0x91 | Byte Offset: 0x244 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_1: Base address MSB for Bottom Frame surface 1 in Memory (for Interlaced Video) This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.49 VI_CSI_1_SURFACE1_BF_OFFSET_LSB_0

Offset: 0x92 | Byte Offset: 0x248 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_1: Base address LSB for Bottom Frame surface 1 in Memory (for Interlaced Video) This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.50 VI_CSI_1_SURFACE2_BF_OFFSET_MSB_0

Offset: 0x93 | Byte Offset: 0x24c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1:0	X	BF_OFFSET_MSB_2: Base address MSB for Bottom Frame surface 2 in Memory (for Interlaced Video) This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.51 VI_CSI_1_SURFACE2_BF_OFFSET_LSB_0

Offset: 0x94 | Byte Offset: 0x250 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BF_OFFSET_LSB_2: Base address LSB for Bottom Frame surface 2 in Memory (for Interlaced Video) This Register field is double buffered. Only atom aligned (64Bytes) address is supported.

31.7.52 VI_CSI_1_SURFACE0_STRIDE_0

Offset: 0x95 | Byte Offset: 0x254 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_0: Stride for Surface 0 in Memory This Register field is double buffered. Only atom aligned (64Bytes) strides are supported.

31.7.53 VI_CSI_1_SURFACE1_STRIDE_0

Offset: 0x96 | Byte Offset: 0x258 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_1: Stride for Surface 1 in Memory This Register field is double buffered. Only atom aligned (64Bytes) strides are supported.

31.7.54 VI_CSI_1_SURFACE2_STRIDE_0

Offset: 0x97 | Byte Offset: 0x25c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	STRIDE_2: Stride for Surface 2 in Memory This Register field is double buffered. Only atom aligned (64Bytes) strides are supported.

31.7.55 VI_CSI_1_SURFACE_HEIGHT0_0

Offset: 0x98 | Byte Offset: 0x260 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3:0	0x1	GOBS: Number of GOBs stacked vertically to form a block

31.7.56 VI_CSI_1_ISPINTF_CONFIG_0

Offset: 0x99 | Byte Offset: 0x264 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx011)

Bit	Reset	Description
2:1	0x1	CHROMA_POS: Stream format 0 = MPEG1 1 = MPEG2
0	0x1	DO_YUV_INTERP: DO_YUV_INTERP: 0=Bypass Chroma Interpolator for YUV 1=Run YUV through Chroma Interpolator

31.7.57 VI_CSI_1_ERROR_STATUS_0

Offset: 0xa1 | Byte Offset: 0x284 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5	X	WATCHDOG_INT: This event occurs when Watchdog timer expires and the EOF is not received on the CSI2VI interface
4	X	CSI_FRAME_ERROR: Flagged if EOF field from CSI has FRAME_ERROR bit set
3	X	FRAME_HEIGHT_LONG_ERROR: Flagged if frame height is larger than HEIGHT. Write 1 to clear.
2	X	FRAME_HEIGHT_SHORT_ERROR: Flagged if frame height is smaller than HEIGHT. Write 1 to clear.
1	X	LINE_WIDTH_LONG_ERROR: Flagged if frame line width is larger than WIDTH. Write 1 to clear.
0	X	LINE_WIDTH_SHORT_ERROR: Flagged if frame line width is smaller than WIDTH. Write 1 to clear.

31.7.58 VI_CSI_1_ERROR_INT_MASK_0

Offset: 0xa2 | Byte Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	WATCHDOG_INT_MASK: Watchdog Timer 0 Interrupt Mask This controls interrupt when WATCHDOG trigger event occurs for the CSI stream 0 = Disabled, 1 = Enabled
4	0x0	CSI_FRAME_ERROR_INT_MASK: CSI error interrupt mask
3	0x0	FRAME_HEIGHT_LONG_INT_MASK: frame height long error interrupt mask
2	0x0	FRAME_HEIGHT_SHORT_INT_MASK: frame height short error interrupt mask
1	0x0	LINE_WIDTH_LONG_INT_MASK: line width long error interrupt mask
0	0x0	LINE_WIDTH_SHORT_INT_MASK: line width short error interrupt mask

31.7.59 VI_CSI_1_WD_CTRL_0

Watch Dog Timer Control Register

Offset: 0xa3 | Byte Offset: 0x28c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	WD_ENABLE: Watch Dog Timer Enable 0 = DISABLE 1 = ENABLE

31.7.60 VI_CSI_1_WD_PERIOD_0

Watch Dog Timer Control Register

Offset: 0xa4 | Byte Offset: 0x290 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	WD_PERIOD: Watch Dog Timer Period in pixel clocks

31.8 VI I2C Registers

31.8.1 VII2C_VI_I2C_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	VI_I2C_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = VI_I2C_ERECOVERY_TX_DONE 5 = VI_I2C_DIRECT_TX_DONE 8 = VI_I2C_DIRECT_RX_DONE 9 = VI_I2C_CSI_PPA_FS 10 = VI_I2C_CSI_PPB_FS 11 = VI_I2C_CSI_PPA_FE 12 = VI_I2C_CSI_PPB_FE 13 = VI_I2C_CSI_PPA_LS 14 = VI_I2C_CSI_PPB_LS 15 = VI_I2C_CSI1_PPA_FS 16 = VI_I2C_CSI1_PPB_FS 17 = VI_I2C_CSI1_PPA_FE 18 = VI_I2C_CSI1_PPB_FE 19 = VI_I2C_CSI1_PPA_LS 20 = VI_I2C_CSI1_PPB_LS 21 = VI_I2C_CSI2_PPA_FS 22 = VI_I2C_CSI2_PPB_FS 23 = VI_I2C_CSI2_PPA_FE 24 = VI_I2C_CSI2_PPB_FE 25 = VI_I2C_CSI2_PPA_LS 26 = VI_I2C_CSI2_PPB_LS 27 = COND_27 28 = COND_28 29 = COND_29 30 = COND_30 31 = COND_31 32 = COND_32
7:0	0x0	VI_I2C_INDX: syncpt index value

31.8.2 VII2C_VI_I2C_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	VI_I2C_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	VI_I2C_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

31.8.3 VII2C_VI_I2C_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VI_I2C_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

31.8.4 VII2C_CTXSW_0

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switching works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS. Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

31.8.5 VII2C_INTSTATUS_0

Offset: 0xb | Byte Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
29	X	SLV_ACK_WITHHELD
28	X	SLV_RD2WR
27	X	SLV_WR2RD
26	X	SLV_PKT_XFER_ERR
25	X	SLV_TX_BUFFER_REQ
24	X	SLV_RX_BUFFER_FILLED
23	X	SLV_PACKET_XFER_COMPLETE
22	X	SLV_TFIFO_OVF
21	X	SLV_RFIFO_UNF
20	X	SLV_TFIFO_DATA_REQ
19	X	SLV_RFIFO_DATA_REQ
18	X	BUS_CLEAR_DONE
17	X	TLOW_MEXT_TIMEOUT
16	X	TLOW_SEXT_TIMEOUT
15	X	TIMEOUT
14	X	PACKET_XFER_COMPLETE
13	X	ALL_PACKETS_XFER_COMPLETE
12	X	TFIFO_OVF
11	X	RFIFO_UNF
10	X	NOACK
9	X	ARB_LOST
8	X	TFIFO_DATA_REQ

Bit	Reset	Description
7	X	RFIFO_DATA_REQ
4	0x0	DIRECT_FIFO_OVERFLOW_ERROR
3	0x0	ERECOVERY_FIFO_OVERFLOW_ERROR
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

31.8.6 VII2C_INT_MASK_0

Offset: 0xc | Byte Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
3	0x0	DIRECT_FIFO_OVERFLOW_ERROR_MASK
2	0x0	ERECOVERY_FIFO_OVERFLOW_ERROR_MASK

31.8.7 VII2C_I2C_IP_INTSTATUS_0

Offset: 0xd | Byte Offset: 0x34 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:0	X	I2C_INT_VALUES

31.8.8 VII2C_ERECOVERY_ERROR_MASK_0

Offset: 0xe | Byte Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000000000000000)

Bit	Reset	Description
29	0x0	SLV_ACK_WITHHELD
28	0x0	SLV_RD2WR
27	0x0	SLV_WR2RD
26	0x0	SLV_PKT_XFER_ERR
25	0x0	SLV_TX_BUFFER_REQ
24	0x0	SLV_RX_BUFFER_FILLED
23	0x0	SLV_PACKET_XFER_COMPLETE
22	0x0	SLV_TFIFO_OVF
21	0x0	SLV_RFIFO_UNF
20	0x0	SLV_TFIFO_DATA_REQ
19	0x0	SLV_RFIFO_DATA_REQ
18	0x0	BUS_CLEAR_DONE
17	0x0	TLOW_MEXT_TIMEOUT
16	0x0	TLOW_SEXT_TIMEOUT
15	0x0	TIMEOUT
14	0x0	PACKET_XFER_COMPLETE
13	0x0	ALL_PACKETS_XFER_COMPLETE
12	0x0	TFIFO_OVF
11	0x0	RFIFO_UNF
10	0x0	NOACK
9	0x0	ARB_LOST
8	0x0	TFIFO_DATA_REQ
7	0x0	RFIFO_DATA_REQ
4	0x0	DIRECT_FIFO_OVERFLOW_ERROR
3	0x0	ERECOVERY_FIFO_OVERFLOW_ERROR
0	0x0	CTXSW_INT

31.8.9 VII2C_STREAM_DIRECT_FENCE_STATUS_0

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

31.8.10 VII2C_STREAM_DIRECT_INIT_FENCE_STATUS_0

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

31.8.11 VII2C_STREAM_ERECOVERY_FENCE_STATUS_0

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

31.8.12 VII2C_STREAM_ERECOVERY_INIT_FENCE_STATUS_0

Offset: 0x16 | Byte Offset: 0x58 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

31.8.13 VII2C_I2C_RESERVED_0_0

Offset: 0x17 | Byte Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

31.8.14 VII2C_I2C_RESERVED_1_0

Offset: 0x18 | Byte Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

31.8.15 VII2C_I2C_STREAM_DIRECT_I2C_CNFG_0

IC Controller Configuration Register (Master)

The I2C_CNFG register is used to configure:

- the number of bytes to be transmitted or received
- the slave device type, either a 7-bit device or a 10-bit device
- Enable mode to send Start Byte or not
- select either a single slave transaction or two slave transaction
- Enable mode to handle devices that do not generate ACK.

Offset: 0x300 | Read/Write: R/W | Reset: 0x00020800 (0bxxxxxxxxxxxxxxx100000100000000000)

Bit	Reset	Description
17	ENABLE	MULTI_MASTER_MODE:used to select single or multi master mode 0=single master; 1=multi_master in single_master mode, no Arbitration checks would happen 0 = DISABLE 1 = ENABLE
15	0x0	MSTR_CLR_BUS_ON_TIMEOUT: When this bit is set, the I2C master will force the clock low for an extended period of time(>TIMEOUT) to force all SMB slaves to release the bus 0 = Do not clear the bus on TLow:SEXT/TIMEOUT timeout 1 = clear the bus on TLow:SEXT/TIMEOUT timeout
14:12	0x0	DEBOUNCE_CNT: Debounce period for sda and scl lines 0 = No debounce, 1 = 2T, 2 = 4T, 3 = 6T, etc., where T is the period of the fix PLL clk source coming to I2C. Maximum debounce period programmable is 14T.A debounce period of >50ns is desirable
11	0x1	NEW_MASTER_FSM: Maintained for compatibility purposes. 0 = DISABLE 1 = ENABLE
10	0x0	PACKET_MODE_EN: Write 1 to initiate transfer in packet mode. 0 = NOP 1 = GO
9	0x0	SEND: Writing a 1 causes the master to initiate the transaction in normal mode. Values of other bits are not affected when this bit is 1,Cleared by hardware. Other bits of the register are masked for writes when this bit is programmed to one.hence,firmware should first configure all other registers and bits [8:0] of I2C_CNFG register before the bit I2C_CNFG[9] is programmed to zero. 0 = NOP 1 = GO
8	0x0	NOACK: Enable mode to handle devices that do not generate ACK. 1 - Do not look for an ack at the end of the Enable 0 = DISABLE 1 = ENABLE
7	0x0	CMD2: Read/Write Command for Slave 2: 1 - Read Transaction; 0 - write Transaction. For a 7-bit slave address,this bit must match with the LSB of address byte for slave 2. Valid only when bit 4 of this register is set 0 = DISABLE 1 = ENABLE
6	0x0	CMD1: Read/Write Command for Slave 1: 1 - Read Transaction; 0 - write Transaction. Command for Slave 1: For a 7-bit slave address this bit must match with the LSB of address byte for slave1. 0 = DISABLE 1 = ENABLE
5	0x0	START: 1 = Yes, a Start byte needs to be sent. 0 = DISABLE 1 = ENABLE
4	0x0	SLV2: 1 - Enables a two slave transaction ; 0 = No command for Slave 2 present. 0 = DISABLE 1 = ENABLE
3:1	0x0	LENGTH: The number of bytes to be transmitted per transaction 000= 1 byte ... 111 = 8 bytes. In a two slave transaction number of bytes should be programmed less than 011.
0	0x0	A_MOD: Address mode defines whether a 7-bit or a 10-bit slave address is programmed. 1 = 10-bit device address, 0 = 7-bit device address 0 = SEVEN_BIT_DEVICE_ADDRESS 1 = TEN_BIT_DEVICE_ADDRESS

31.8.16 VII2C_I2C_STREAM_DIRECT_I2C_CMD_ADDR0_0

I2C Slave-1 Address

I2C_CMD_ADDR0 is programmed the 7 Bit or 10 Bit address of slave 1 with which the transaction is intended;

Offset: 0x304 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR0: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[6]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[6] indicates the read/write transaction.

31.8.17 VII2C_I2C_STREAM_DIRECT_I2C_CMD_ADDR1_0

I2C Slave-2 Address

I2C_CMD_ADDR1 is programmed the 7 Bit or 10 Bit address of slave 2 with which the transaction is intended;

Offset: 0x308 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR1: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[7]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[7] indicates the read/write transaction.

31.8.18 VII2C_I2C_STREAM_DIRECT_I2C_CMD_DATA1_0

IC Controller Data 1: Transmit/Receive

The four Least Significant Bytes of Data to be Transmitted is loaded into the register when I2c Master is in Write Mode; the four Least Significant Bytes of Data are Read through this register when I2c Master is in Read mode.

Offset: 0x30c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA4: Fourth data byte to be sent/received
23:16	0x0	DATA3: Third data byte to be sent/received
15:8	0x0	DATA2: Second data byte to be sent/received
7:0	0x0	DATA1: This register contains the first data byte to be sent/received.

31.8.19 VII2C_I2C_STREAM_DIRECT_I2C_CMD_DATA2_0

IC Controller Data 2: Transmit/Receive

The four Most Significant Bytes of Data to be transmitted is loaded into the register when I2c Master is in Write Mode; the four Most Significant Bytes of Data are Read through this register when I2c Master is in Read mode.

Offset: 0x310 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA8: Eighth data byte to be sent/received
23:16	0x0	DATA7: Seventh data byte to be sent/received
15:8	0x0	DATA6: Sixth data byte to be sent/received
7:0	0x0	DATA5: This register contains the Fifth data byte to be sent/received.

31.8.20 VII2C_I2C_STREAM_DIRECT_I2C_STATUS_0

IC Controller Status (Master)

I2C_STATUS gives the status of I2c Master operation

Offset: 0x31c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	BUSY: 1 = Busy. 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
7:4	X	CMD2_STAT: Transaction for Slave2 for x byte failed. x is 'h0 to 'ha, all others invalid. 0 = SL2_XFER_SUCCESSFUL 1 = SL2_NOACK_FOR_BYTE1 2 = SL2_NOACK_FOR_BYTE2 3 = SL2_NOACK_FOR_BYTE3 4 = SL2_NOACK_FOR_BYTE4 5 = SL2_NOACK_FOR_BYTE5 6 = SL2_NOACK_FOR_BYTE6 7 = SL2_NOACK_FOR_BYTE7 8 = SL2_NOACK_FOR_BYTE8 9 = SL2_NOACK_FOR_BYTE9 10 = SL2_NOACK_FOR_BYTE10
3:0	X	CMD1_STAT: Transaction for Slave1 for x byte failed. x is 'h0 to 'ha, all others invalid. 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3 4 = SL1_NOACK_FOR_BYTE4 5 = SL1_NOACK_FOR_BYTE5 6 = SL1_NOACK_FOR_BYTE6 7 = SL1_NOACK_FOR_BYTE7 8 = SL1_NOACK_FOR_BYTE8 9 = SL1_NOACK_FOR_BYTE9 10 = SL1_NOACK_FOR_BYTE10

31.8.21 VII2C_I2C_STREAM_DIRECT_I2C_SL_CNFG_0

IC Controller Configuration (Slave)

I2C_SL_CNFG register is used to configure, Enable mode of slave Ack, and Enable mode of slave response to general call address. The register should be programmed when I2c controller is configured as slave.

Offset: 0x320 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxx00000000000000000100)

Bit	Reset	Description
20	0x0	FIFO_XFER_EN: If this bit is disabled,data is always communicated via I2C_SL_RCVD register. If enabled, it is through FIFOs 0 = DISABLE 1 = ENABLE
19:8	0x0	BUFFER_SIZE: Payload size in bytes
7	0x0	ACK_LAST_BYTE_VALID:ack the last byte valid (Write-Only). This bit qualifies ACK_LAST_BYTE field 0 = DISABLE 1 = ENABLE
6	0x0	ACK_LAST_BYTE:ack the last byte 0 = DISABLE 1 = ENABLE
5	0x0	ACK_WITHHOLD_EN: Ack Withhold Feature Enable 0 = DISABLE 1 = ENABLE
4	0x0	PKT_MODE_EN: Packet Mode Enable 0 = DISABLE 1 = ENABLE
3	0x0	ENABLE_SL: By writing zero to this field,slave can be turned off 0 = DISABLE 1 = ENABLE
2	0x1	NEWSL: New Slave 1 - use new slave 0 = DISABLE 1 = ENABLE
1	0x0	NACK: Disable Slave Ack. 1 - slave will not ack reception of address or data byte. 0 = DISABLE 1 = ENABLE
0	0x0	RESP: Slave response to general call address (zero address) 1 - Enable. 0 = DISABLE 1 = ENABLE

31.8.22 VII2C_I2C_STREAM_DIRECT_I2C_SL_RCVD_0

IC Controller Slave Receive/Transmit Data (Slave)

Offset: 0x324 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SL_DATA:slave Received data

31.8.23 VII2C_I2C_STREAM_DIRECT_I2C_SL_STATUS_0

IC Controller Slave Status (Slave)

Offset: 0x328 | Read/Write: R/W | Reset: 0x0000XX0X (0bxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	R/W	Reset	Description
14:8	RO	X	HW_MSTR_ADR: HW master addr received via general call addressing. This field is meaningful only if HW_MSTR_INT is set.
7	RW	0x0	HW_MSTR_INT: 1 = Interrupt has been generated by slave Hardware Master Address is received after General Call Address. 1 = Received HW Master Address 0 = No event.
6	RW	0x0	REPROG_SL: 1 = Interrupt has been generated by slave By after General Call Address is 0x04. 1 = Reprogram slave address. 0 = No action.
5	RW	0x0	RST_SL: 1 = Interrupt has been generated by slave By after General Call Address is 0x06. 1 = Reset and reprogram slave address. 0 = No action.
4	RW	0x0	END_TRANS: 1 = Interrupt has been generated by slave Transaction completed as indicated by stop/repeat start condition. 1 = Transaction completed. 0 = No transaction occurred or transaction in progress.
3	RW	0x0	SL_IRQ: 1 = Interrupt has been generated by slave 0 = No interrupt generated 0 = UNSET 1 = SET
2	RW	0x0	RCVD: New Transaction Received status 1 = Transaction occurred. 0 = No transaction occurred 0 = NO_TRANSACTION_OCCURED 1 = TRANSACTION_OCCURED
1	RO	X	RNW: Slave Transaction status 0 = Write 1=Read 0 = WRITE 1 = READ
0	RW	0x0	ZA: Zero Address Status 1 = Yes, slave responded 0 = No, slave did not respond 0 = NO_SLAVE_RESPONSE 1 = SLAVE_RESPONSE

31.8.24 VII2C_I2C_STREAM_DIRECT_I2C_SL_ADDR1_0

IC Controller Slave Address 1 Register (Slave)

Offset: 0x32c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	SL_ADDR1: For a 10-bit slave address, this field is the least significant 8 bits.
7:0	0x0	SL_ADDR0: For a 10-bit slave address, this field is the least significant 8 bits.

31.8.25 VII2C_I2C_STREAM_DIRECT_I2C_SL_ADDR2_0

IC Controller Slave Address 2 Register (Slave)

Offset: 0x330 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx0xxxxx000xxxxx000)

Bit	Reset	Description
16	0x0	SELECT_SLAVE: 0 = Use slave addr0 1 = Use slave addr1
10:9	0x0	SL1_ADDR_HI: In 7 bit address mode these bits are don't care; In 10 bit address mode they represent the 2 MSB of the address.

Bit	Reset	Description
8	0x0	SL1_VLD: 0 = 7-bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE
2:1	0x0	SL_ADDR_HI: In 7 bit address mode these bits are don't care; In 10 bit address mode they represent the 2 MSB of the address.
0	0x0	VLD: 0 = 7-bit addressing. 1 - 10 bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE

31.8.26 VII2C_I2C_STREAM_DIRECT_I2C_TLOW_SEXT_0

IC Controller SMBUS timeout thresholds

Offset: 0x334 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27	0x0	RST_SL_ON_TIMEOUT: Reset Slave state machine on time-out
26	0x0	TLOW_MEXT_EN: Enable TLOW_MEXT counter
25	0x0	TLOW_SEXT_EN: Enable TLOW_SEXT counter
24	0x0	TIMEOUT_EN: Enable TIMEOUT counter
23:16	0x0	TLOW_MEXT: cumulative clock low extend time(master device) accumulated over a byte transfer period in milliseconds(START to ACK,ACK to ACK, or ACK to STOP).
15:8	0x0	TLOW_SEXT:cumulative clock low extend time(slave device) accumulated over a complete transfer(START till STOP)
7:0	0x0	TIMEOUT:clock low timeout period in milliseconds

31.8.27 VII2C_I2C_STREAM_DIRECT_I2C_SL_DELAY_COUNT_0

IC Slave Controller Delay Count

Offset: 0x33c | Read/Write: R/W | Reset: 0x0000001e (0bxxxxxxxxxxxxxxxx0000000000011110)

Bit	Reset	Description
15:0	0x1e	SL_DELAY_COUNT: The value determines the timing between an address cycle and a subsequent data cycle or two consecutive data cycles on the bus. The I2C_SL_DELAY_COUNT is valid only when internal slave is accessed. I2C_SL_DELAY_COUNT has to be programmed such that TIMING = T * DLY where T is period of clock source selected for I2c; and DLY is I2C_SL_DELAY_COUNT ; TIMING is the desired timing, A value of >= 1250 ns is advisable.

31.8.28 VII2C_I2C_STREAM_DIRECT_I2C_SL_INT_MASK_0

IC Controller Slave Mask (Slave)

Offset: 0x340 | Read/Write: R/W | Reset: 0x000000fd (0bxxxxxxxxxxxxxxxxxxxxxxxxx111111x1)

Bit	Reset	Description
7	0x1	HW_MSTR_INT: 0 = DISABLE 1 = ENABLE
6	0x1	REPROG_SL: 0 = DISABLE 1 = ENABLE
5	0x1	RST_SL: 0 = DISABLE 1 = ENABLE
4	0x1	END_TRANS: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x1	SL_IRQ: 0 = DISABLE 1 = ENABLE
2	0x1	RCVD: 0 = DISABLE 1 = ENABLE
0	0x1	ZA: 0 = DISABLE 1 = ENABLE

31.8.29 VII2C_I2C_STREAM_DIRECT_I2C_SL_INT_SOURCE_0

IC Controller Slave Source (Slave)

Offset: 0x344 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	HW_MSTR_INT: 0 = UNSET 1 = SET
6	X	REPROG_SL: 0 = UNSET 1 = SET
5	X	RST_SL: 0 = UNSET 1 = SET
4	X	END_TRANS: 0 = UNSET 1 = SET
3	X	SL_IRQ: 0 = UNSET 1 = SET
2	X	RCVD: 0 = UNSET 1 = SET
0	X	ZA: 0 = UNSET 1 = SET

31.8.30 VII2C_I2C_STREAM_DIRECT_I2C_SL_INT_SET_0

IC Controller Slave Source (Slave)

Offset: 0x348 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	Reset	Description
7	0x0	HW_MSTR_INT: 0 = UNSET 1 = SET
6	0x0	REPROG_SL: 0 = UNSET 1 = SET
5	0x0	RST_SL: 0 = UNSET 1 = SET
4	0x0	END_TRANS: 0 = UNSET 1 = SET
3	0x0	SL_IRQ: 0 = UNSET 1 = SET

Bit	Reset	Description
2	0x0	RCVD: 0 = UNSET 1 = SET
0	0x0	ZA: 0 = UNSET 1 = SET

Note: *Program the field PACKET_MODE_EN of I2C_CNFG register while working in packet mode.*

The set of registers below describe the interface for packet mode only. These registers describe the interface for normal mode. When the packet mode interface changes, normal mode registers and the operation are not affected. The transition from normal mode to packet mode is suggested because packet mode allows

1. There is no restriction on the number of bytes before and the after the repeated start
2. The number of bytes that can be transferred with a single cmd is not limited to 8.
   Though a packet can contain 4k bytes,because any number of packets can be pushed into the FIFO,there is no limit on the number of bytes that can be transferred.
3. The transactions to different slaves can be chained together with repeat start and there is no limit on the number of slaves it can address.

31.8.31 VII2C_I2C_STREAM_DIRECT_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. The header size is variable and could vary from 2 to 5 words. For I2C, it is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction specific information. Payload contains actual data to be written to the slave. In case of read operation, Payload is nil,hence the packet contains header only .

Offset: 0x350 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET:Software writes packets into this register. A packet may contain a generic header or I2C specific header or data.

31.8.32 VII2C_I2C_STREAM_DIRECT_I2C_RX_FIFO_0

Offset: 0x354 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: Software reads of data from this register,cause a pop

31.8.33 VII2C_I2C_STREAM_DIRECT_PACKET_TRANSFER_STATUS_0

Offset: 0x358 | Read/Write: RO | Reset: 0x0XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	TRANSFER_COMPLETE:The packet transfer for which last packet is set has been completed 0 = UNSET 1 = SET
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet
3	X	NOACK_FOR_ADDR:No ack received for the address byte 0 = UNSET 1 = SET
2	X	NOACK_FOR_DATA:No ack received for the data byte 0 = UNSET 1 = SET

Bit	Reset	Description
1	X	ARB_LOST:Arbitration lost for the current byte 0 = UNSET 1 = SET
0	X	CONTROLLER_BUSY:1 = Controller is busy 0 = UNSET 1 = SET

31.8.34 VII2C_I2C_STREAM_DIRECT_FIFO_CONTROL_0

Offset: 0x35c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:13	0x0	SLV_TX_FIFO_TRIG:Slave Transmit FIFO trigger level 000 = 1 word, DMA trigger is asserted when at least one word is empty in the FIFO 001 = 2 word, DMA trigger is asserted when at least 2 words are empty in the FIFO
12:10	0x0	SLV_RX_FIFO_TRIG:Slave Receive FIFO trigger level 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
9	0x0	SLV_TX_FIFO_FLUSH:1= flush the Tx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET
8	0x0	SLV_RX_FIFO_FLUSH:1= flush the Rx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET
7:5	0x0	TX_FIFO_TRIG:Transmit FIFO trigger level 000 = 1 word, DMA trigger is asserted when at least one word is empty in the FIFO 001 = 2 word, DMA trigger is asserted when at least 2 words are empty in the FIFO
4:2	0x0	RX_FIFO_TRIG:Receive FIFO trigger level 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
1	0x0	TX_FIFO_FLUSH:1= flush the Tx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET
0	0x0	RX_FIFO_FLUSH:1= flush the Rx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET

31.8.35 VII2C_I2C_STREAM_DIRECT_FIFO_STATUS_0

Offset: 0x360 | Read/Write: RO | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	SLV_XFER_ERR_REASON:This bit describes the nature of the packet transfer error.It is meaningful only if PKT_XFER_ERR is set 0 = Master terminated transaction before it was completed 1 = Master did not terminate transaction when all bytes are transferred
23:20	X	SLV_TX_FIFO_EMPTY_CNT:The number of slots that can be written to the slave Tx FIFO 0000 = tx_fifo full 0001 = 1 slot empty 0010 = 2 slots empty
19:16	X	SLV_RX_FIFO_FULL_CNT:The number of slots to be read from the Slave Rx FIFO 0000 = rx_fifo empty 0001 = 1 slot full 0010 = 2 slots full
7:4	X	TX_FIFO_EMPTY_CNT:The number of slots that can be written to the Tx FIFO 0000 = tx_fifo full 0001 = 1 slot empty 0010 = 2 slots empty

Bit	Reset	Description
3:0	X	RX_FIFO_FULL_CNT:The number of slots to be read from the Rx FIFO 0000 = rx_fifo empty 0001 = 1 slot full 0010 = 2 slots full

31.8.36 VII2C_I2C_STREAM_DIRECT_INTERRUPT_MASK_REGISTER_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xx00xxxx000000000000)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD_INT_EN: 0 = DISABLE 1 = ENABLE
27	0x0	SLV_RD2WR_INT_EN: 0 = DISABLE 1 = ENABLE
26	0x0	SLV_WR2RD_INT_EN: 0 = DISABLE 1 = ENABLE
25	0x0	SLV_PKT_XFER_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
24	0x0	SLV_TX_BUFFER_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
23	0x0	SLV_RX_BUFFER_FILLED_INT_EN: 0 = DISABLE 1 = ENABLE
22	0x0	SLV_PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
21	0x0	SLV_TFIFO_OVF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
20	0x0	SLV_RFIFO_UNF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
17	0x0	SLV_TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
16	0x0	SLV_RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
11	0x0	BUS_CLEAR_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
10	0x0	TLOW_MEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
9	0x0	TLOW_SEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
8	0x0	TIMEOUT_INT_EN: 0 = DISABLE 1 = ENABLE
7	0x0	PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	TFIFO_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RFIFO_UNF_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	NOACK_INT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	ARB_LOST_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE

31.8.37 VII2C_I2C_STREAM_DIRECT_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set. If set,Write 1 to clear it However TFIFO_DATA_REQ,RFIFO_DATA_REQ fields depend on the FIFO trigger levels and cannot be cleared.

Slv_rd2wr indicates there is a switch from rd to wr by repeat start and the current rd transaction needs to be closed and start with wr transaction.Similarly slv_wr2rd indicates switch from wr to rd.

Offset: 0x368 | Read/Write: R/W | Reset: 0x000X000X (0bxxx000000000xxxxxxxx0000000000xx)

Bit	R/W	Reset	Description
28	RW	0x0	SLV_ACK_WITHHELD:ack is withheld,waiting for software explicit information about ack 0 = UNSET 1 = SET
27	RW	0x0	SLV_RD2WR:Transaction switching from rd to wr 0 = UNSET 1 = SET
26	RW	0x0	SLV_WR2RD:Transaction switching from wr to rd 0 = UNSET 1 = SET
25	RW	0x0	SLV_PKT_XFER_ERR:0 = Request was successful 1 = Error has occurred during packet transfer 0 = UNSET 1 = SET
24	RW	0x0	SLV_TX_BUFFER_REQ:slave Tx buffer is full 0 = UNSET 1 = SET
23	RW	0x0	SLV_RX_BUFFER_FILLED:slave Rx buffer is full 0 = UNSET 1 = SET
22	RW	0x0	SLV_PACKET_XFER_COMPLETE:slave packet transfer complete 0 = UNSET 1 = SET
21	RW	0x0	SLV_TFIFO_OVF:slave Tx FIFO overflow 0 = UNSET 1 = SET
20	RW	0x0	SLV_RFIFO_UNF:slave Rx FIFO underflow 0 = UNSET 1 = SET
17	RO	X	SLV_TFIFO_DATA_REQ:slave Tx FIFO data req 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
16	RO	X	SLV_RFIFO_DATA_REQ:slave Rx FIFO data req 0 = UNSET 1 = SET
11	RW	0x0	BUS_CLEAR_DONE:bus clear done status 0 = UNSET 1 = SET
10	RW	0x0	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	RW	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	RW	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	RW	0x0	PACKET_XFER_COMPLETE:A packet has been transferred successfully.TRANSFER_PKT_ID field can be used to know the current byte under transfer.This bit can be masked by the IE field in the I2C specific header 0 = UNSET 1 = SET
6	RW	0x0	ALL_PACKETS_XFER_COMPLETE:All the packets transferred successfully 0 = UNSET 1 = SET
5	RW	0x0	TFIFO_OVF: Tx FIFO overflow 0 = UNSET 1 = SET
4	RW	0x0	RFIFO_UNF: Rx FIFO underflow 0 = UNSET 1 = SET
3	RW	0x0	NOACK:No ACK from slave 0 = UNSET 1 = SET
2	RW	0x0	ARB_LOST:Arbitration lost 0 = UNSET 1 = SET
1	RO	X	TFIFO_DATA_REQ: Tx FIFO data req 0 = UNSET 1 = SET
0	RO	X	RFIFO_DATA_REQ: Rx FIFO data req 0 = UNSET 1 = SET

31.8.38 VII2C_I2C_STREAM_DIRECT_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- scl freq (std/fast/fm+ modes) = ClkSourceFreq/((tlow+thigh+3)*(N+1)) for lower values of N, up to 3
- scl freq (std/fast/fm+ modes) = ClkSourceFreq/((tlow+thigh+2)*(N+1)) for higher values of N, above 3
- scl freq (HS mode) = ClkSourceFreq/((ths_low+ths_high+4)*(N+1)) for lower values of N, up to 4
- scl freq (HS mode) = ClkSourceFreq/((ths_low+ths_high+2)*(N+1)) for higher values of N, above 4

Offset: 0x36c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000110010000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE:N= divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE:N= divide by n+1

31.8.39 VII2C_I2C_STREAM_DIRECT_I2C_INTERRUPT_SOURCE_REGISTER_0

This is a read-only register which returns the AND of Interrupt Source and Interrupt Mask registers.

Offset: 0x370 | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	X	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	X	SLV_RD2WR: 0 = UNSET 1 = SET
26	X	SLV_WR2RD: 0 = UNSET 1 = SET
25	X	SLV_PKT_XFER_ERR:Error occurred during slave transfer 0 = UNSET 1 = SET
24	X	SLV_TX_BUFFER_REQ:slave Tx buffer is full 0 = UNSET 1 = SET
23	X	SLV_RX_BUFFER_FILLED:slave Rx buffer is full 0 = UNSET 1 = SET
22	X	SLV_PACKET_XFER_COMPLETE:slave packet transfer complete 0 = UNSET 1 = SET
21	X	SLV_TFIFO_OVF:slave Tx FIFO overflow 0 = UNSET 1 = SET
20	X	SLV_RFIFO_UNF:slave Rx FIFO underflow 0 = UNSET 1 = SET
17	X	SLV_TFIFO_DATA_REQ:slave Tx FIFO data req 0 = UNSET 1 = SET
16	X	SLV_RFIFO_DATA_REQ:slave Rx FIFO data req 0 = UNSET 1 = SET
11	X	BUS_CLEAR_DONE:bus clear done 0 = UNSET 1 = SET
10	X	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	X	TLOW_SEXT_TIMEOUT: SMBUS sext time-out 0 = UNSET 1 = SET
8	X	TIMEOUT: SMBUS time-out 0 = UNSET 1 = SET
7	X	PACKET_XFER_COMPLETE: packet transferred successfully 0 = UNSET 1 = SET
6	X	ALL_PACKETS_XFER_COMPLETE:All the packets transferred successfully 0 = UNSET 1 = SET
5	X	TFIFO_OVF: Tx FIFO overflow 0 = UNSET 1 = SET
4	X	RFIFO_UNF: Rx FIFO underflow 0 = UNSET 1 = SET
3	X	NOACK:No ACK from slave 0 = UNSET 1 = SET

Bit	Reset	Description
2	X	ARB_LOST:Arbitration lost 0 = UNSET 1 = SET
1	X	TFIFO_DATA_REQ: Tx FIFO data req 0 = UNSET 1 = SET
0	X	RFIFO_DATA_REQ: Rx FIFO data req 0 = UNSET 1 = SET

31.8.40 VII2C_I2C_STREAM_DIRECT_I2C_INTERRUPT_SET_REGISTER_0

This is a write-only register which can be used to set the interrupt status bit. A write to this register causes the bits in status register to be set if the corresponding bit in write data is 1'b1. Read always returns 'h0.

Offset: 0x374 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xxxxxxxx0000000000xx)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	0x0	SLV_RD2WR: 0 = UNSET 1 = SET
26	0x0	SLV_WR2RD: 0 = UNSET 1 = SET
25	0x0	SLV_PKT_XFER_ERR:Error occurred during slave transfer 0 = UNSET 1 = SET
24	0x0	SLV_TX_BUFFER_REQ:slave Tx buffer is full 0 = UNSET 1 = SET
23	0x0	SLV_RX_BUFFER_FILLED:slave Rx buffer is full 0 = UNSET 1 = SET
22	0x0	SLV_PACKET_XFER_COMPLETE:slave packet transfer complete 0 = UNSET 1 = SET
21	0x0	SLV_TFIFO_OVF:slave Tx FIFO overflow 0 = UNSET 1 = SET
20	0x0	SLV_RFIFO_UNF:slave Rx FIFO underflow 0 = UNSET 1 = SET
11	0x0	BUS_CLEAR_DONE:bus clear done 0 = UNSET 1 = SET
10	0x0	TLOW_MEXT_TIMEOUT:SMBUS mext time-out 0 = UNSET 1 = SET
9	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	0x0	PACKET_XFER_COMPLETE: packet transferred successfully 0 = UNSET 1 = SET

Bit	Reset	Description
6	0x0	ALL_PACKETS_XFER_COMPLETE:All the packets transferred successfully 0 = UNSET 1 = SET
5	0x0	TFIFO_OVF: Tx FIFO overflow 0 = UNSET 1 = SET
4	0x0	RFIFO_UNF: Rx FIFO underflow 0 = UNSET 1 = SET
3	0x0	NOACK:No ACK from slave 0 = UNSET 1 = SET
2	0x0	ARB_LOST:Arbitration lost 0 = UNSET 1 = SET

31.8.41 VII2C_I2C_STREAM_DIRECT_I2C_SLV_TX_PACKET_FIFO_0

Offset: 0x378 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET:Software writes packets into this register. A packet may contain a generic header or I2C specific header or data.

31.8.42 VII2C_I2C_STREAM_DIRECT_I2C_SLV_RX_FIFO_0

Offset: 0x37c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA:Software reads of data from this register,cause a pop

31.8.43 VII2C_I2C_STREAM_DIRECT_I2C_SLV_PACKET_STATUS_0

Offset: 0x380 | Read/Write: RO | Reset: 0x0XXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
25	X	ACK_WITHHELD:Indicates that ack is withheld for last byte and slave is waiting for the host to explicitly command slave to ACK the last byte 0 = Bus is released 1 = ACK is withheld
24	X	TRANSFER_COMPLETE:ALL the packets have been transferred successfully
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet

31.8.44 VII2C_I2C_STREAM_DIRECT_I2C_BUS_CLEAR_CONFIG_0

Offset: 0x384 | Read/Write: R/W | Reset: 0x00090004 (0bxxxxxxxx00001001xxxxxxxxxxxx100)

Bit	Reset	Description
23:16	0x9	BC_SCLK_THRESHOLD:send the clock pulses until this threshold is met
2	0x1	BC_STOP_COND: 0 = NO_STOP : do not send stop condition at the end of bus clear operation 1 = STOP : send stop condition at the end of the bus clear operation
1	0x0	BC_TERMINATE: 0 = THRESHOLD : irrespective of SDA release status during BC, terminate the BC only after threshold is reached. 1 = IMMEDIATE : terminate the bus clear operation immediately when SDA is released or threshold count is reached whichever is earlier
0	0x0	BC_ENABLE: starts bus clear operation, HW auto-clears this bit upon bus clear transaction completion

31.8.45 VII2C_I2C_STREAM_DIRECT_I2C_BUS_CLEAR_STATUS_0

Offset: 0x388 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	BC_STATUS: 0 = NOT_CLEARED : indicates SDA is not released by slave, its status is still low. 1 = CLEARED : SDA is released

31.8.46 VII2C_I2C_STREAM_DIRECT_I2C_CONFIG_LOAD_0

Offset: 0x38c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	TIMEOUT_CONFIG_LOAD: This bit loads the timeout configuration from pclk domain to the receive (i2c_slow_clk)domain. Software has to set this bit finally after doing the required registers configuration for the logic to take the updates. Once the internal update is done, hardware auto-clears this bit. Since the hardware would be busy with internal updating, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE
1	0x0	SLV_CONFIG_LOAD:This bit loads the slave configuration from pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration for the slave controller to take updates. Once the internal update is done, hardware auto-clears this bit. Since the hardware would be busy with internal updating, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE
0	0x0	MSTR_CONFIG_LOAD:This bit loads the master configuration from pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration like I2C_I2C_CNFG_0 bit fields etc. Once the internal update is done, hardware auto-clears this bit. Since the hardware would be busy with internal updating, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE

31.8.47 VII2C_I2C_STREAM_DIRECT_I2C_CLKEN_OVERRIDE_0

Offset: 0x390 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000x0)

Bit	Reset	Description
4	CLK_GATED	I2C_BUS_CLEAR_CLKEN_OVR: override for 2nd-level clock enable for I2C bus clear logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON
3	CLK_GATED	I2C_SLV_HIF_CLKEN_OVR: override for 2nd-level clock enable for I2C slave to host interface logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON
2	CLK_GATED	I2C_SLV_CORE_CLKEN_OVR: override for 2nd-level clock enable for I2C slave core logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON
0	CLK_GATED	I2C_MST_CORE_CLKEN_OVR: override for 2nd-level clock enable for I2C master core logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON

31.8.48 VII2C_I2C_STREAM_DIRECT_I2C_INTERFACE_TIMING_0_0

register for Standard/Fast/Fm+ mode timing

Offset: 0x394 | Read/Write: R/W | Reset: 0x00000204 (0bxxxxxxxxxxxxxxxxxx000010xx000100)

Bit	Reset	Description
13:8	0x2	THIGH: High period of the SCL clock
5:0	0x4	TLOW: Low period of the SCL clock

31.8.49 VII2C_I2C_STREAM_DIRECT_I2C_INTERFACE_TIMING_1_0

register for Standard/Fast/Fm+ mode timing

Offset: 0x398 | Read/Write: R/W | Reset: 0x04070404 (0bxx000100xx000111xx000100xx000100)

Bit	Reset	Description
29:24	0x4	TBUF:Bus free time between STOP and START conditions
21:16	0x7	TSU_STO:Setup time for STOP condition
13:8	0x4	THD_STA: Hold time for a (repeated) START condition
5:0	0x4	TSU_STA:Setup time for a Repeated START condition

31.8.50 VII2C_I2C_STREAM_DIRECT_I2C_HS_INTERFACE_TIMING_0_0

I2C interface timing register for HS mode transfers Since HS master code is sent in Std/Fm/FMm+ modes, standard/fast/fm+ timing registers need to be programmed too along with HS timing registers

Offset: 0x39c | Read/Write: R/W | Reset: 0x00000308 (0bxxxxxxxxxxxxxxxxxxxx000011xx001000)

Bit	Reset	Description
13:8	0x3	HS_THIGH:High period of the SCL clock
5:0	0x8	HS_TLOW:Low period of the SCL clock

31.8.51 VII2C_I2C_STREAM_DIRECT_I2C_HS_INTERFACE_TIMING_1_0

I2C interface timing register for HS mode transfers. Since HS master code is sent in Std/Fm/FMm+ modes, standard/fast/fm+ timing registers need to be programmed too along with HS timing registers

Offset: 0x3a0 | Read/Write: R/W | Reset: 0x000b0b0b (0bxxxxxxxxxx001011xx001011xx001011)

Bit	Reset	Description
21:16	0xb	HS_TSU_STO:Setup time for STOP condition
13:8	0xb	HS_THD_STA: Hold time for a (repeated) START condition
5:0	0xb	HS_TSU_STA: Setup time for a Repeated START condition

31.8.52 VII2C_I2C_STREAM_ERECOVERY_I2C_CNFG_0

IC Controller Configuration Register (Master)

The I2C_CNFG register is used to configure:

- the number of bytes to be transmitted or received
- the slave device type, either a 7-bit device or a 10-bit device
- Enable mode to send Start Byte or not
- select either a single slave transaction or two slave transaction
- Enable mode to handle devices that do not generate ACK.

Offset: 0x400 | Read/Write: R/W | Reset: 0x00020800 (0bxxxxxxxxxxxxxxx100000100000000000)

Bit	Reset	Description
17	ENABLE	MULTI_MASTER_MODE:used to select single or multi master mode 0=single master; 1=multi_master in single_master mode, no Arbitration checks would happen 0 = DISABLE 1 = ENABLE
15	0x0	MSTR_CLR_BUS_ON_TIMEOUT: When this bit is set, the I2C master will force the clock low for an extended period of time(>TIMEOUT) to force all SMB slaves to release the bus 0 = Do not clear the bus on TLow:SEXT/TIMEOUT timeout 1 = clear the bus on TLow:SEXT/TIMEOUT timeout

Bit	Reset	Description
14:12	0x0	DEBOUNCE_CNT: Debounce period for sda and scl lines 0 = No debounce, 1 = 2T, 2 = 4T, 3 = 6T, etc., where T is the period of the fix PLL clk source coming to I2C. Maximum debounce period programmable is 14T.A debounce period of >50ns is desirable
11	0x1	NEW_MASTER_FSM: Maintained for compatibility purposes. 0 = DISABLE 1 = ENABLE
10	0x0	PACKET_MODE_EN: Write 1 to initiate transfer in packet mode. 0 = NOP 1 = GO
9	0x0	SEND: Writing a 1 causes the master to initiate the transaction in normal mode. Values of other bits are not affected when this bit is 1,Cleared by hardware. Other bits of the register are masked for writes when this bit is programmed to one.hence,firmware should first configure all other registers and bits [8:0] of I2C_CNFG register before the bit I2C_CNFG[9] is programmed to zero. 0 = NOP 1 = GO
8	0x0	NOACK: Enable mode to handle devices that do not generate ACK. 1 - do not look for an ack at the end of the Enable 0 = DISABLE 1 = ENABLE
7	0x0	CMD2: Read/Write Command for Slave 2: 1 - Read Transaction; 0 - write Transaction. For a 7-bit slave address,this bit must match with the LSB of address byte for slave 2. Valid only when bit 4 of this register is set 0 = DISABLE 1 = ENABLE
6	0x0	CMD1: Read/Write Command for Slave 1: 1 - Read Transaction; 0 - write Transaction. Command for Slave 1: For a 7-bit slave address this bit must match with the LSB of address byte for slave1. 0 = DISABLE 1 = ENABLE
5	0x0	START: 1 = Yes, a Start byte needs to be sent. 0 = DISABLE 1 = ENABLE
4	0x0	SLV2: 1 - Enables a two slave transaction ; 0 = No command for Slave 2 present. 0 = DISABLE 1 = ENABLE
3:1	0x0	LENGTH: The Number of bytes to be transmitted per transaction 000= 1 byte ... 111 = 8 bytes. In a two slave transaction number of bytes should be programmed less than 011.
0	0x0	A_MOD: Address mode defines whether a 7-bit or a 10-bit slave address is programmed. 1 = 10-bit device address, 0 = 7-bit device address 0 = SEVEN_BIT_DEVICE_ADDRESS 1 = TEN_BIT_DEVICE_ADDRESS

31.8.53 VII2C_I2C_STREAM_ERECOVERY_I2C_CMD_ADDR0_0

I2C Slave-1 Address

I2C_CMD_ADDR0 is programmed the 7 Bit or 10 Bit address of slave 1 with which the transaction is intended;

Offset: 0x404 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR0: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[6]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[6] indicates the read/write transaction.

31.8.54 VII2C_I2C_STREAM_ERECOVERY_I2C_CMD_ADDR1_0

I2C Slave-2 Address

I2C_CMD_ADDR1 is programmed the 7 Bit or 10 Bit address of slave 2 with which the transaction is intended;

Offset: 0x408 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR1: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[7]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[7] indicates the read/write transaction.

31.8.55 VII2C_I2C_STREAM_ERECOVERY_I2C_CMD_DATA1_0

IC Controller Data 1: Transmit/Receive

The four Least Significant Bytes of Data to be Transmitted is loaded into the register when I2c Master is in Write Mode;

The four Least Significant Bytes of Data are Read through this register when I2c Master is in Read mode.

Offset: 0x40c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA4: Fourth data byte to be sent/received
23:16	0x0	DATA3: Third data byte to be sent/received
15:8	0x0	DATA2: Second data byte to be sent/received
7:0	0x0	DATA1: This register contains the first data byte to be sent/received.

31.8.56 VII2C_I2C_STREAM_ERECOVERY_I2C_CMD_DATA2_0

IC Controller Data 2: Transmit/Receive

The four Most Significant Bytes of Data to be transmitted is loaded into the register when I2c Master is in Write Mode; The four Most Significant Bytes of Data are Read through this register when I2c Master is in Read mode.

Offset: 0x410 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA8: Eighth data byte to be sent/received
23:16	0x0	DATA7: Seventh data byte to be sent/received
15:8	0x0	DATA6: Sixth data byte to be sent/received
7:0	0x0	DATA5: This register contains the Fifth data byte to be sent/received.

31.8.57 VII2C_I2C_STREAM_ERECOVERY_I2C_STATUS_0

IC Controller Status (Master)

I2C_STATUS gives the status of I2c Master operation

Offset: 0x41c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	BUSY: 1 = Busy. 0 = NOT_BUSY 1 = BUSY
7:4	X	CMD2_STAT: Transaction for Slave2 for x byte failed. x is 'h0 to 'ha, all others invalid. 0 = SL2_XFER_SUCCESSFUL 1 = SL2_NOACK_FOR_BYTE1 2 = SL2_NOACK_FOR_BYTE2 3 = SL2_NOACK_FOR_BYTE3 4 = SL2_NOACK_FOR_BYTE4 5 = SL2_NOACK_FOR_BYTE5 6 = SL2_NOACK_FOR_BYTE6 7 = SL2_NOACK_FOR_BYTE7 8 = SL2_NOACK_FOR_BYTE8 9 = SL2_NOACK_FOR_BYTE9 10 = SL2_NOACK_FOR_BYTE10

Bit	Reset	Description
3:0	X	CMD1_STAT: Transaction for Slave1 for x byte failed. x is 'h0 to 'ha, all others invalid. 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3 4 = SL1_NOACK_FOR_BYTE4 5 = SL1_NOACK_FOR_BYTE5 6 = SL1_NOACK_FOR_BYTE6 7 = SL1_NOACK_FOR_BYTE7 8 = SL1_NOACK_FOR_BYTE8 9 = SL1_NOACK_FOR_BYTE9 10 = SL1_NOACK_FOR_BYTE10

31.8.58 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_CNFG_0

IC Controller Configuration (Slave)

I2C_SL_CNFG register is used to configure, Enable mode of slave Ack,and Enable mode of slave response to general call address. The register should be programmed when I2c controller is configured as slave.

Offset: 0x420 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxx00000000000000000100)

Bit	Reset	Description
20	0x0	FIFO_XFER_EN: If this bit is disabled,data is always communicated via I2C_SL_RCVD register. If enabled, it is through FIFOs 0 = DISABLE 1 = ENABLE
19:8	0x0	BUFFER_SIZE: Payload size in bytes
7	0x0	ACK_LAST_BYTE_VALID:ack the last byte valid(Write-Only). This bit qualifies ACK_LAST_BYTE field 0 = DISABLE 1 = ENABLE
6	0x0	ACK_LAST_BYTE:ack the last byte 0 = DISABLE 1 = ENABLE
5	0x0	ACK_WITHHOLD_EN: Ack Withhold Feature Enable 0 = DISABLE 1 = ENABLE
4	0x0	PKT_MODE_EN: Packet Mode Enable 0 = DISABLE 1 = ENABLE
3	0x0	ENABLE_SL: By writing zero to this field,slave can be turned off 0 = DISABLE 1 = ENABLE
2	0x1	NEWSL: New Slave 1 - use new slave 0 = DISABLE 1 = ENABLE
1	0x0	NACK: Disable Slave Ack. 1 - slave will not ack reception of address or data byte. 0 = DISABLE 1 = ENABLE
0	0x0	RESP: Slave response to general call address (zero address) 1 - Enable. 0 = DISABLE 1 = ENABLE

31.8.59 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_RCVD_0

IC Controller Slave Receive/Transmit Data (Slave)

Offset: 0x424 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SL_DATA:slave Received data

31.8.60 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_STATUS_0

IC Controller Slave Status (Slave)

Offset: 0x428 | Read/Write: R/W | Reset: 0x0000XX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	R/W	Reset	Description
14:8	RO	X	HW_MSTR_ADR: HW master addr received via general call addressing. This field is meaningful only if HW_MSTR_INT is set.
7	RW	0x0	HW_MSTR_INT: 1 = Interrupt has been generated by slave Hardware Master Address is received after General Call Address. 1 = Received HW Master Address 0 = No event.
6	RW	0x0	REPROG_SL: 1 = Interrupt has been generated by slave By after General Call Address is 0x04. 1 = Reprogram slave address. 0 = No action.
5	RW	0x0	RST_SL: 1 = Interrupt has been generated by slave By after General Call Address is 0x06. 1 = Reset and reprogram slave address. 0 = No action.
4	RW	0x0	END_TRANS: 1 = Interrupt has been generated by slave Transaction completed as indicated by stop/repeat start condition. 1 = Transaction completed. 0 = No transaction occurred or transaction in progress.
3	RW	0x0	SL_IRQ: 1 = Interrupt has been generated by slave 0 = No interrupt generated 0 = UNSET 1 = SET
2	RW	0x0	RCVD: New Transaction Received status 1 = Transaction occurred. 0 = No transaction occurred 0 = NO_TRANSACTION_OCCURED 1 = TRANSACTION_OCCURED
1	RO	X	RNW: Slave Transaction status 0 = Write 1=Read 0 = WRITE 1 = READ
0	RW	0x0	ZA: Zero Address Status 1 = Yes, slave responded 0 = No, slave did not respond 0 = NO_SLAVE_RESPONSE 1 = SLAVE_RESPONSE

31.8.61 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_ADDR1_0

IC Controller Slave Address 1 Register (Slave)

Offset: 0x42c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	SL_ADDR1: For a 10-bit slave address, this field is the least significant 8 bits.
7:0	0x0	SL_ADDR0: For a 10-bit slave address, this field is the least significant 8 bits.

31.8.62 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_ADDR2_0

IC Controller Slave Address 2 Register (Slave)

Offset: 0x430 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxx000xxxxx000)

Bit	Reset	Description
16	0x0	SELECT_SLAVE: 0 = Use slave addr0 1 = Use slave addr1
10:9	0x0	SL1_ADDR_HI: In 7 bit address mode these bits are don't care; In 10 bit address mode they represent the 2 MSB of the address.
8	0x0	SL1_VLD: 0 = 7-bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE
2:1	0x0	SL_ADDR_HI: In 7 bit address mode these bits are don't care; In 10 bit address mode they represent the 2 MSB of the address.
0	0x0	VLD: 0 = 7-bit addressing. 1 - 10 bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE

31.8.63 VII2C_I2C_STREAM_ERECOVERY_I2C_TLOW_SEXT_0

IC Controller SMBUS timeout thresholds

Offset: 0x434 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27	0x0	RST_SL_ON_TIMEOUT: Reset Slave state machine on time-out
26	0x0	TLOW_MEXT_EN: Enable TLOW_MEXT counter
25	0x0	TLOW_SEXT_EN: Enable TLOW_SEXT counter
24	0x0	TIMEOUT_EN: Enable TIMEOUT counter
23:16	0x0	TLOW_MEXT: cumulative clock low extend time(master device) accumulated over a byte transfer period in milliseconds(START to ACK,ACK to ACK, or ACK to STOP).
15:8	0x0	TLOW_SEXT:cumulative clock low extend time(slave device) accumulated over a complete transfer(START till STOP)
7:0	0x0	TIMEOUT:clock low timeout period in milliseconds

31.8.64 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_DELAY_COUNT_0

IC Slave Controller Delay Count

Offset: 0x43c | Read/Write: R/W | Reset: 0x0000001e (0bxxxxxxxxxxxxxxxx0000000000011110)

Bit	Reset	Description
15:0	0x1e	SL_DELAY_COUNT: The value determines the timing between an address cycle and a subsequent data cycle or two consecutive data cycles on the bus. The I2C_SL_DELAY_COUNT is valid only when internal slave is accessed. I2C_SL_DELAY_COUNT has to be programmed such that TIMING = T * DLY where T is period of clock source selected for I2c; and DLY is I2C_SL_DELAY_COUNT ; TIMING is the desired timing, A value of >= 1250 ns is advisable.

31.8.65 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_INT_MASK_0

IC Controller Slave Mask (Slave)

Offset: 0x440 | Read/Write: R/W | Reset: 0x000000fd (0bxxxxxxxxxxxxxxxxxxxxxxxxxx111111x1)

Bit	Reset	Description
7	0x1	HW_MSTR_INT: 0 = DISABLE 1 = ENABLE
6	0x1	REPROG_SL: 0 = DISABLE 1 = ENABLE
5	0x1	RST_SL: 0 = DISABLE 1 = ENABLE
4	0x1	END_TRANS: 0 = DISABLE 1 = ENABLE
3	0x1	SL_IRQ: 0 = DISABLE 1 = ENABLE
2	0x1	RCVD: 0 = DISABLE 1 = ENABLE
0	0x1	ZA: 0 = DISABLE 1 = ENABLE

31.8.66 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_INT_SOURCE_0

IC Controller Slave Source (Slave)

Offset: 0x444 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	HW_MSTR_INT: 0 = UNSET 1 = SET
6	X	REPROG_SL: 0 = UNSET 1 = SET
5	X	RST_SL: 0 = UNSET 1 = SET
4	X	END_TRANS: 0 = UNSET 1 = SET
3	X	SL_IRQ: 0 = UNSET 1 = SET
2	X	RCVD: 0 = UNSET 1 = SET
0	X	ZA: 0 = UNSET 1 = SET

31.8.67 VII2C_I2C_STREAM_ERECOVERY_I2C_SL_INT_SET_0

IC Controller Slave Source (Slave)

Offset: 0x448 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	Reset	Description
7	0x0	HW_MSTR_INT: 0 = UNSET 1 = SET
6	0x0	REPROG_SL: 0 = UNSET 1 = SET
5	0x0	RST_SL: 0 = UNSET 1 = SET
4	0x0	END_TRANS: 0 = UNSET 1 = SET
3	0x0	SL_IRQ: 0 = UNSET 1 = SET
2	0x0	RCVD: 0 = UNSET 1 = SET
0	0x0	ZA: 0 = UNSET 1 = SET

Note: *Program the field PACKET_MODE_EN of I2C_CNFG register while working in packet mode.*

The set of registers below describe the interface for packet mode only. These registers describe the interface for normal mode. When the packet mode interface changes, normal mode registers and the operation are not affected. The transition from normal mode to packet mode is suggested because packet mode allows

1. There is no restriction on the number of bytes before and the after the repeated start
2. The number of bytes that can be transferred with a single cmd is not limited to 8.
   Though a packet can contain 4kbytes because any number of packets can be pushed into the FIFO, there is no limit on the number of bytes that can be transferred.
3. The transactions to different slaves can be chained together with repeat start and there is no limit on the number of slaves it can address.

31.8.68 VII2C_I2C_STREAM_ERECOVERY_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. The header size is variable and could vary from 2 to 5 words. For I2C, it is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction specific information.payload contains actual data to be written to the slave. In case of read operation, payload is nil,hence the packet contains header only .

Offset: 0x450 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET:Software writes packets into this register. A packet may contain a generic header or I2C specific header or data.

31.8.69 VII2C_I2C_STREAM_ERECOVERY_I2C_RX_FIFO_0

Offset: 0x454 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA:SW Reads data from this register,causes pop

31.8.70 VII2C_I2C_STREAM_ERECOVERY_PACKET_TRANSFER_STATUS_0

Offset: 0x458 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	TRANSFER_COMPLETE:The packet transfer for which last packet is set has been completed 0 = UNSET 1 = SET
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM:The number of bytes transferred in the current packet
3	X	NOACK_FOR_ADDR:No ack received for the address byte 0 = UNSET 1 = SET
2	X	NOACK_FOR_DATA:No ack received for the data byte 0 = UNSET 1 = SET
1	X	ARB_LOST:Arbitration lost for the current byte 0 = UNSET 1 = SET
0	X	CONTROLLER_BUSY:1 = Controller is busy 0 = UNSET 1 = SET

31.8.71 VII2C_I2C_STREAM_ERECOVERY_FIFO_CONTROL_0

Offset: 0x45c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:13	0x0	SLV_TX_FIFO_TRIG:Slave Transmit FIFO trigger level 000 = 1 word, DMA trigger is asserted when at least one word is empty in the FIFO 001 = 2 word, DMA trigger is asserted when at least 2 words are empty in the FIFO
12:10	0x0	SLV_RX_FIFO_TRIG:Slave Receive FIFO trigger level 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
9	0x0	SLV_TX_FIFO_FLUSH:1= flush the Tx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET
8	0x0	SLV_RX_FIFO_FLUSH:1= flush the Rx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET
7:5	0x0	TX_FIFO_TRIG:Transmit FIFO trigger level 000 = 1 word, DMA trigger is asserted when at least one word is empty in the FIFO 001 = 2 word, DMA trigger is asserted when at least 2 words are empty in the FIFO
4:2	0x0	RX_FIFO_TRIG:Receive FIFO trigger level 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
1	0x0	TX_FIFO_FLUSH:1= flush the Tx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET
0	0x0	RX_FIFO_FLUSH:1= flush the Rx FIFO, cleared after the FIFO is flushed 0 = UNSET 1 = SET

31.8.72 VII2C_I2C_STREAM_ERECOVERY_FIFO_STATUS_0

Offset: 0x460 | Read/Write: RO | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	SLV_XFER_ERR_REASON: This bit describes the nature of the packet transfer error. It is meaningful only if PKT_XFER_ERR is set 0 = Master terminated transaction before it was completed 1 = Master did not terminate transaction when all bytes are transferred
23:20	X	SLV_TX_FIFO_EMPTY_CNT:The number of slots that can be written to the slave Tx FIFO 0000 = tx_fifo full 0001 = 1 slot empty 0010 = 2 slots empty
19:16	X	SLV_RX_FIFO_FULL_CNT:The number of slots to be read from the Slave Rx FIFO 0000 = rx_fifo empty 0001 = 1 slot full 0010 = 2 slots full
7:4	X	TX_FIFO_EMPTY_CNT:The number of slots that can be written to the Tx FIFO 0000 = tx_fifo full 0001 = 1 slot empty 0010 = 2 slots empty
3:0	X	RX_FIFO_FULL_CNT:The number of slots to be read from the Rx FIFO 0000 = rx_fifo empty 0001 = 1 slot full 0010 = 2 slots full

31.8.73 VII2C_I2C_STREAM_ERECOVERY_INTERRUPT_MASK_REGISTER_0

Offset: 0x464 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xx00xxxx000000000000)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD_INT_EN: 0 = DISABLE 1 = ENABLE
27	0x0	SLV_RD2WR_INT_EN: 0 = DISABLE 1 = ENABLE
26	0x0	SLV_WR2RD_INT_EN: 0 = DISABLE 1 = ENABLE
25	0x0	SLV_PKT_XFER_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
24	0x0	SLV_TX_BUFFER_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
23	0x0	SLV_RX_BUFFER_FILLED_INT_EN: 0 = DISABLE 1 = ENABLE
22	0x0	SLV_PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
21	0x0	SLV_TFIFO_OVF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
20	0x0	SLV_RFIFO_UNF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
17	0x0	SLV_TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
16	0x0	SLV_RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
11	0x0	BUS_CLEAR_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
10	0x0	TLOW_MEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
9	0x0	TLOW_SEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
8	0x0	TIMEOUT_INT_EN: 0 = DISABLE 1 = ENABLE
7	0x0	PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
5	0x0	TFIFO_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RFIFO_UNF_INT_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x0	NOACK_INT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	ARB_LOST_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE

31.8.74 VII2C_I2C_STREAM_ERECOVERY_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set.If set, write 1 to clear it; however, TFIFO_DATA_REQ,RFIFO_DATA_REQ fields depend on the FIFO trigger levels and cannot be cleared.

Slv_rd2wr indicates there is a switch from rd to wr by repeat start and the current rd transaction needs to be closed and start with wr transaction. Similarly slv_wr2rd indicates switch from wr to rd.

Offset: 0x468 | Read/Write: R/W | Reset: 0x000X000X (0bxxx000000000xxxxxxxx0000000000xx)

Bit	R/W	Reset	Description
28	RW	0x0	SLV_ACK_WITHHELD:ack is withheld,waiting for software explicit information about ack 0 = UNSET 1 = SET
27	RW	0x0	SLV_RD2WR:Transaction switching from rd to wr 0 = UNSET 1 = SET
26	RW	0x0	SLV_WR2RD:Transaction switching from wr to rd 0 = UNSET 1 = SET
25	RW	0x0	SLV_PKT_XFER_ERR:0 = Request was successful 1 = Error has occurred during packet transfer 0 = UNSET 1 = SET
24	RW	0x0	SLV_TX_BUFFER_REQ:slave Tx buffer is full 0 = UNSET 1 = SET
23	RW	0x0	SLV_RX_BUFFER_FILLED:slave Rx buffer is full 0 = UNSET 1 = SET
22	RW	0x0	SLV_PACKET_XFER_COMPLETE:slave packet transfer complete 0 = UNSET 1 = SET
21	RW	0x0	SLV_TFIFO_OVF:slave Tx FIFO overflow 0 = UNSET 1 = SET
20	RW	0x0	SLV_RFIFO_UNF:slave Rx FIFO underflow 0 = UNSET 1 = SET
17	RO	X	SLV_TFIFO_DATA_REQ:slave Tx FIFO data req 0 = UNSET 1 = SET
16	RO	X	SLV_RFIFO_DATA_REQ:slave Rx FIFO data req 0 = UNSET 1 = SET
11	RW	0x0	BUS_CLEAR_DONE:bus clear done status 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
10	RW	0x0	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	RW	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	RW	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	RW	0x0	PACKET_XFER_COMPLETE:A packet has been transferred successfully.TRANSFER_PKT_ID field can be used to know the current byte under transfer. This bit can be masked by the IE field in the I2C specific header 0 = UNSET 1 = SET
6	RW	0x0	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	RW	0x0	TFIFO_OVF: Tx FIFO overflow 0 = UNSET 1 = SET
4	RW	0x0	RFIFO_UNF: Rx FIFO underflow 0 = UNSET 1 = SET
3	RW	0x0	NOACK:No ACK from slave 0 = UNSET 1 = SET
2	RW	0x0	ARB_LOST:Arbitration lost 0 = UNSET 1 = SET
1	RO	X	TFIFO_DATA_REQ: Tx FIFO data req 0 = UNSET 1 = SET
0	RO	X	RFIFO_DATA_REQ: Rx FIFO data req 0 = UNSET 1 = SET

31.8.75 VII2C_I2C_STREAM_ERECOVERY_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- scl freq (std/fast/fm+ modes) = ClkSourceFreq/((tlow+thigh+3)*(N+1)) for lower values of N, up to 3
- scl freq (std/fast/fm+ modes) = ClkSourceFreq/((tlow+thigh+2)*(N+1)) for higher values of N, above 3
- scl freq (HS mode) = ClkSourceFreq/((ths_low+ths_high+4)*(N+1)) for lower values of N, up to 4
- scl freq (HS mode) = ClkSourceFreq/((ths_low+ths_high+2)*(N+1)) for higher values of N, above 4

Offset: 0x46c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000110010000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE:N= divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE:N= divide by n+1

31.8.76 VII2C_I2C_STREAM_ERECOVERY_I2C_INTERRUPT_SOURCE_REGISTER_0

This is a read-only register which returns the AND of Interrupt Source and Interrupt Mask registers.

Offset: 0x470 | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	X	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	X	SLV_RD2WR: 0 = UNSET 1 = SET
26	X	SLV_WR2RD: 0 = UNSET 1 = SET
25	X	SLV_PKT_XFER_ERR:Error occurred during slave transfer 0 = UNSET 1 = SET
24	X	SLV_TX_BUFFER_REQ:slave Tx buffer is full 0 = UNSET 1 = SET
23	X	SLV_RX_BUFFER_FILLED:slave Rx buffer is full 0 = UNSET 1 = SET
22	X	SLV_PACKET_XFER_COMPLETE:slave packet transfer complete 0 = UNSET 1 = SET
21	X	SLV_TFIFO_OVF:slave Tx FIFO overflow 0 = UNSET 1 = SET
20	X	SLV_RFIFO_UNF:slave Rx FIFO underflow 0 = UNSET 1 = SET
17	X	SLV_TFIFO_DATA_REQ:slave Tx FIFO data req 0 = UNSET 1 = SET
16	X	SLV_RFIFO_DATA_REQ:slave Rx FIFO data req 0 = UNSET 1 = SET
11	X	BUS_CLEAR_DONE:bus clear done 0 = UNSET 1 = SET
10	X	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	X	TLOW_SEXT_TIMEOUT: SMBUS sext time-out 0 = UNSET 1 = SET
8	X	TIMEOUT: SMBUS time-out 0 = UNSET 1 = SET
7	X	PACKET_XFER_COMPLETE: packet transferred successfully 0 = UNSET 1 = SET
6	X	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	X	TFIFO_OVF: Tx FIFO overflow 0 = UNSET 1 = SET
4	X	RFIFO_UNF: Rx FIFO underflow 0 = UNSET 1 = SET
3	X	NOACK:No ACK from slave 0 = UNSET 1 = SET

Bit	Reset	Description
2	X	ARB_LOST:Arbitration lost 0 = UNSET 1 = SET
1	X	TFIFO_DATA_REQ: Tx FIFO data req 0 = UNSET 1 = SET
0	X	RFIFO_DATA_REQ: Rx FIFO data req 0 = UNSET 1 = SET

31.8.77 VII2C_I2C_STREAM_ERECOVERY_I2C_INTERRUPT_SET_REGISTER_0

This is a write-only register which can be used to set the interrupt status bit. A write to this register causes the bits in status register to be set if the corresponding bit in write data is 1'b1. Read always returns 'h0.

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xxxxxxxx0000000000xx)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	0x0	SLV_RD2WR: 0 = UNSET 1 = SET
26	0x0	SLV_WR2RD: 0 = UNSET 1 = SET
25	0x0	SLV_PKT_XFER_ERR:Error occurred during slave transfer 0 = UNSET 1 = SET
24	0x0	SLV_TX_BUFFER_REQ:slave Tx buffer is full 0 = UNSET 1 = SET
23	0x0	SLV_RX_BUFFER_FILLED:slave Rx buffer is full 0 = UNSET 1 = SET
22	0x0	SLV_PACKET_XFER_COMPLETE:slave packet transfer complete 0 = UNSET 1 = SET
21	0x0	SLV_TFIFO_OVF:slave Tx FIFO overflow 0 = UNSET 1 = SET
20	0x0	SLV_RFIFO_UNF:slave Rx FIFO underflow 0 = UNSET 1 = SET
11	0x0	BUS_CLEAR_DONE:bus clear done 0 = UNSET 1 = SET
10	0x0	TLOW_MEXT_TIMEOUT:SMBUS mext time-out 0 = UNSET 1 = SET
9	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	0x0	PACKET_XFER_COMPLETE: packet transferred successfully 0 = UNSET 1 = SET

Bit	Reset	Description
6	0x0	ALL_PACKETS_XFER_COMPLETE:All the packets transferred successfully 0 = UNSET 1 = SET
5	0x0	TFIFO_OVF: Tx FIFO overflow 0 = UNSET 1 = SET
4	0x0	RFIFO_UNF: Rx FIFO underflow 0 = UNSET 1 = SET
3	0x0	NOACK:No ACK from slave 0 = UNSET 1 = SET
2	0x0	ARB_LOST:Arbitration lost 0 = UNSET 1 = SET

31.8.78 VII2C_I2C_STREAM_ERECOVERY_I2C_SLV_TX_PACKET_FIFO_0

Offset: 0x478 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET:Software writes packets into this register. A packet may contain a generic header or I2C specific header or data.

31.8.79 VII2C_I2C_STREAM_ERECOVERY_I2C_SLV_RX_FIFO_0

Offset: 0x47c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA:SW Reads data from this register,causes pop

31.8.80 VII2C_I2C_STREAM_ERECOVERY_I2C_SLV_PACKET_STATUS_0

Offset: 0x480 | Read/Write: RO | Reset: 0x0XXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
25	X	ACK_WITHHELD:Indicates that ack is withheld for last byte and slave is waiting for the host to explicitly command slave to ACK the last byte 0 = Bus is released 1 = ACK is withheld
24	X	TRANSFER_COMPLETE:ALL the packets have been transferred successfully
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM:The number of bytes transferred in the current packet

31.8.81 VII2C_I2C_STREAM_ERECOVERY_I2C_BUS_CLEAR_CONFIG_0

Offset: 0x484 | Read/Write: R/W | Reset: 0x00090004 (0bxxxxxxxx00001001xxxxxxxxxxxx100)

Bit	Reset	Description
23:16	0x9	BC_SCLK_THRESHOLD:send the clock pulses until this threshold is met
2	0x1	BC_STOP_COND: 0 = NO_STOP : do not send stop condition at the end of bus clear operation 1 = STOP : send stop condition at the end of the bus clear operation
1	0x0	BC_TERMINATE: 0 = THRESHOLD : irrespective of SDA release status during BC, terminate the BC only after threshold is reached. 1 = IMMEDIATE : terminate the bus clear operation immediately when SDA is released or threshold count is reached whichever is earlier
0	0x0	BC_ENABLE: starts bus clear operation, HW auto-clears this bit upon bus clear transaction completion

31.8.82 VII2C_I2C_STREAM_ERECOVERY_I2C_BUS_CLEAR_STATUS_0

Offset: 0x488 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	BC_STATUS: 0 = NOT_CLEARED : indicates SDA is not released by slave, its status is still low. 1 = CLEARED : SDA is released

31.8.83 VII2C_I2C_STREAM_ERECOVERY_I2C_CONFIG_LOAD_0

Offset: 0x48c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	TIMEOUT_CONFIG_LOAD: This bit loads the timeout configuration from pclk domain to the receive (i2c_slow_clk)domain. Software has to set this bit finally after doing the required registers configuration for the logic to take the updates. Once the internal update is done, hardware auto-clears this bit.Since the hardware would be busy with internal updating, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE
1	0x0	SLV_CONFIG_LOAD: This bit loads the slave configuration from pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration for the slave controller to take updates. Once the internal update is done, hardware auto-clears this bit. Since the hardware would be busy with internal updating, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE
0	0x0	MSTR_CONFIG_LOAD: This bit loads the master configuration from pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration like I2C_I2C_CNFG_0 bit fields etc. Once the internal update is done, hardware auto-clears this bit. Since the hardware would be busy with internal updating, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE

31.8.84 VII2C_I2C_STREAM_ERECOVERY_I2C_CLKEN_OVERRIDE_0

Offset: 0x490 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000x0)

Bit	Reset	Description
4	CLK_GATED	I2C_BUS_CLEAR_CLKEN_OVR: override for 2nd-level clock enable for I2C bus clear logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON
3	CLK_GATED	I2C_SLV_HIF_CLKEN_OVR: override for 2nd-level clock enable for I2C slave to host interface logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON
2	CLK_GATED	I2C_SLV_CORE_CLKEN_OVR: override for 2nd-level clock enable for I2C slave core logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON
0	CLK_GATED	I2C_MST_CORE_CLKEN_OVR: override for 2nd-level clock enable for I2C master core logic 0 = CLK_GATED 1 = CLK_ALWAYS_ON

31.8.85 VII2C_I2C_STREAM_ERECOVERY_I2C_INTERFACE_TIMING_0_0

register for Standard/Fast/Fm+ mode timing

Offset: 0x494 | Read/Write: R/W | Reset: 0x00000204 (0bxxxxxxxxxxxxxxxxxx000010xx000100)

Bit	Reset	Description
13:8	0x2	THIGH:High period of the SCL clock
5:0	0x4	TLOW:Low period of the SCL clock

31.8.86 VII2C_I2C_STREAM_ERECOVERY_I2C_INTERFACE_TIMING_1_0

register for Standard/Fast/Fm+ mode timing

Offset: 0x498 | Read/Write: R/W | Reset: 0x04070404 (0bxx000100xx000111xx000100xx000100)

Bit	Reset	Description
29:24	0x4	TBUF:Bus free time between STOP and START conditions
21:16	0x7	TSU_STO:Setup time for STOP condition
13:8	0x4	THD_STA: Hold time for a (repeated) START condition
5:0	0x4	TSU_STA:Setup time for a Repeated START condition

31.8.87 VII2C_I2C_STREAM_ERECOVERY_I2C_HS_INTERFACE_TIMING_0_0

I2C interface timing register for HS mode transfers. Since HS master code is sent in Std/Fm/FMm+ modes, standard/fast/fm+ timing registers need to be programmed too along with HS timing registers

Offset: 0x49c | Read/Write: R/W | Reset: 0x00000308 (0bxxxxxxxxxxxxxxxxxxx000011xx001000)

Bit	Reset	Description
13:8	0x3	HS_THIGH:High period of the SCL clock
5:0	0x8	HS_TLOW:Low period of the SCL clock

31.8.88 VII2C_I2C_STREAM_ERECOVERY_I2C_HS_INTERFACE_TIMING_1_0

I2C interface timing register for HS mode transfers. Since HS master code is sent in Std/Fm/FMm+ modes, standard/fast/fm+ timing registers need to be programmed too along with HS timing registers

Offset: 0x4a0 | Read/Write: R/W | Reset: 0x000b0b0b (0bxxxxxxxxxx001011xx001011xx001011)

Bit	Reset	Description
21:16	0xb	HS_TSU_STO:Setup time for STOP condition
13:8	0xb	HS_THD_STA: Hold time for a (repeated) START condition
5:0	0xb	HS_TSU_STA:Setup time for a Repeated START condition

31.9 MIPI-CSI Registers

Refer to Chapter 29: MIPI-CSI (Camera Serial Interface) in this document for descriptions of these registers.

CHAPTER 32: SD/MMC CONTROLLER

The Tegra® X1 SD/MMC Controller can interface with SD Cards or eSD, SDIO or eMMC devices. It supports both non-removable devices and plug-in cards. It has a direct MCCIF2 interface and can initiate data transfers between memory and the external card. It has an APB Slave interface to access its configuration registers. To access the iRAM for Micro Boot, the SD/MMC controller relies on the AHB Redirection Controller (ARC) in the Memory Controller.

The SD/MMC controller supports two different bus protocols: SD/SDIO bus protocol and eMMC bus protocol for eMMC devices.

The Tegra X1 device supports 4 SDMMC controllers with 4 pin-muxing options. Each SD/MMC controller is required to talk to one type of device. Each SDMMC controller is capable of all protocols. The combination of SD/MMC controller together with the type of pads it connects to defines the use case for that SD/MMC controller. The use cases for SD/MMC controllers are:

- SDMMC1/SDMMC3: WiFi on SDIO, removable mass storage memory, or removable SD card with 3.3V/1.8V I/O support (Note: This is for primary use case. In general, one can use any slot for SDIO as all slots support interrupt on DAT1 line.)
- SDMMC2: eMMC5.0 device or SDIO with 1.8V I/O support
- SDMMC4: eMMC5.0 device (embedded flash memory) for Booting and Non-volatile storage

32.1 Supported Specifications and Standards

The SD/MMC controller supports the specifications from the SD Card Association and JEDEC (MMC) at the versions listed in the following table.

Table 196: Protocol Versions

Protocol	Version
SD	4.0 (UHS-I only)
SDIO	4.0 (UHS-I only)
eSD	2.1
eMMC	5.0
SDHOST	4.0 (UHS-I only)

The specifications are as follows:

- “SD Specifications, Part E1, SDIO Specification”, Technical Committee SD Card Association, Version 4.00, February 20, 2012
- “SD Specifications, Part 1, Physical Layer Specification”, Technical Committee SD Card Association, Version 4.00, May 30, 2011.
- “SD Specification, Part 1, eSD (Embedded SD) Addendum”, Technical Committee SD Card Association, Version 2.10, November 25, 2008.
- “SD Specifications, Part A2, SD Host Controller Standard Specification”, Technical Committee SD Card Association, Version 4.00, February 20, 2012.
- “JEDEC Standard, Embedded Multi-Media Card (eMMC) Electrical Standard (5.0)”, JEDEC Solid State Technology Association, September 2013.
- “JEDEC standard, Embedded Multi-Media Card (eMMC) Electrical Standard (5.1)”, JEDEC Solid State Technology Association, DRAFT version partly. It is possible to deploy the software version CQE operation, Enhanced Strobe Scheme to avoid tuning mechanism.

32.2 Supported Speeds

These tables show the maximum data rates available over the physical interface. The speeds achievable in actual use will be less than these rates, depending on the performance of the device itself, protocol limitations, and the software driver.

For SD 3.0 data transfer modes, these are the maximum data transfer speeds:

Table 197: Maximum Data Transfer Speeds (SD)

Speed Mode	Signal Voltage	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	UHS Protocol
Default Speed	3.3	25	1,4	12.5	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
HIGH SPEED	3.3	50	1,4	25	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR12	1.8	25	1,4	12.5	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR25	1.8	50	1,4	25	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR50	1.8	100	1,4	50	UHS-I, 50 (SD3.0) UHS-I, 104 (SD3.0)
SDR104 UHS-I	1.8	208	1,4	104	UHS-I, 104 (SD3.0)

For eSD 2.1 data transfer modes, these are the maximum data transfer speeds:

Table 198: Maximum Data Transfer Speeds (eSD 2.1)

Speed Mode	Signal Voltage	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	eSD Protocol
Default Speed	3.3	25	1,4,8	12.5	eSD 2.1
High Speed	3.3	50	1,4,8	25	eSD 2.1

For SDIO data transfer modes, these are the maximum data transfer speeds:

Table 199: Maximum Data Transfer Speeds (SDIO)

Speed Mode	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	Protocol Version/UHS Version
Default Speed	25	1,4	12.5	SDIO2.0 /UHS-I, 50/ UHS-I, 104
High Speed	50	1,4	25	SDIO2.0 /UHS-I, 50/ UHS-I, 104
SDR12	25	1,4	12.5	UHS-I, 50 (SDIO3.0) UHS-I, 104 (SDIO3.0)
SDR25	50	1,4	25	UHS-I, 50 (SDIO3.0) UHS-I, 104 (SDIO3.0)
SDR50	100	1,4	50	UHS-I, 50 (SDIO3.0) UHS-I, 104 (SDIO3.0)
SDR104 UHS-I	208	1,4	104	UHS-I, 104 (SDIO3.0)

For eMMC data transfer modes, these are the maximum data transfer speeds:

Table 200: Maximum Data Transfer Speeds (eMMC)

Speed Mode	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	MMC Revision
LEGACY SPEED	26	1,4,8	26	MMC 4.3
HIGH SPEED SDR	52	1,4,8	52	MMC 4.3
HIGH SPEED DDR	52	4,8	104	MMC 4.4
HS200 (SDR)	200	4,8	200	MMC 4.51
HS400 (DDR)	200	8	400	eMMC5.0

Table 200: Maximum Data Transfer Speeds (eMMC)

Speed Mode	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)	MMC Revision
HS533 (DDR)	266	8	533	Overclocked HS400 mode (Nvidia proprietary)

32.3 Operation

32.3.1 Hardware/Software Partitioning

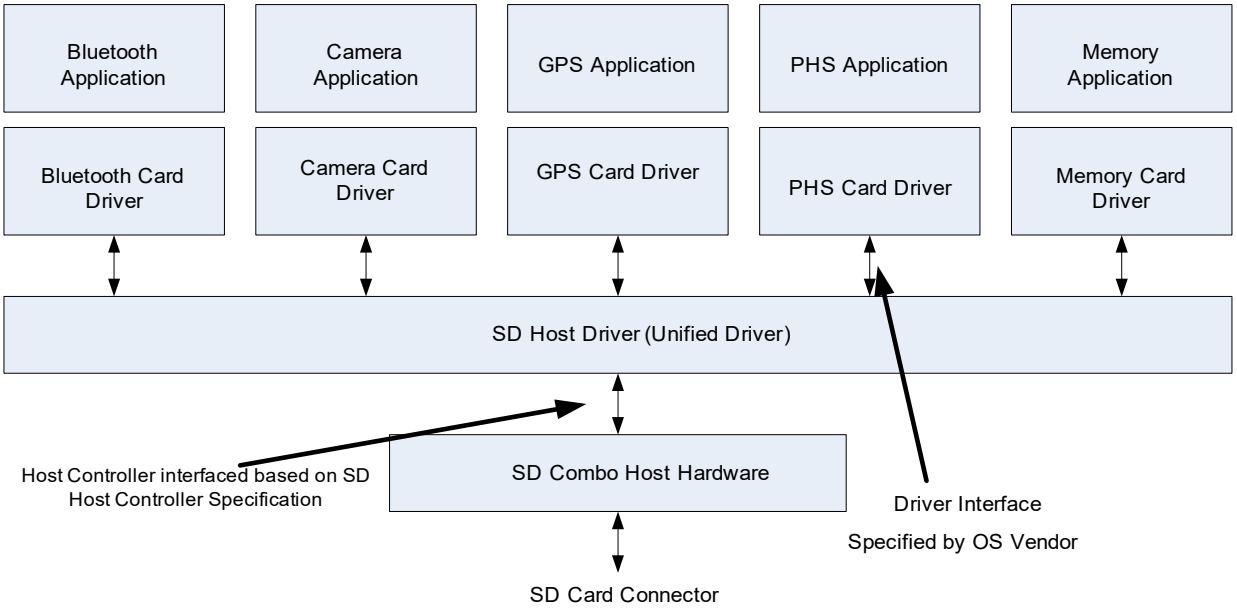
Hardware

The role of the hardware controller is to send out the programmed command, store the response, and make it visible to software. It updates the status registers and generates interrupts when attention is required. Software may also configure it for DMA operation.

Software

The software driver determines which type of card is inserted in the slot by sending the initialization commands and observing the responses received from the card. After identifying the card that is inserted, it should only program the corresponding set of commands that are applicable. It can enable interrupts for which notification is desired.

Figure 147: Host Hardware and Driver Architecture



32.4 Caveats and Assumptions

- A single SD/MMC controller handles only one device at a time.
- The software should configure SDMMC_VENDOR_CLOCK_CNTRL_0_SDMMC_CLK to DISABLE before turning off the SD/MMC clock (and configure back to ENABLE after turning on the SD/MMC clock). This is required in order to support asynchronous card interrupts when no clock is supplied to the card.
  - SDMMC_VENDOR_CLOCK_CNTRL_0_SDMMC_CLK can be left as the Reset Value (ENABLE) if Asynchronous Interrupt is not used.
- PMC can be configured to wake the Tegra X1 device from the LP0 power state based on either:
  - Card Detection pin.
  - Asynchronous interrupt on the DAT1 line.
- Write protect and card detect logic is implemented in Tegra only for SDMMC1 and SDMMC3.

- Off-card ECC, described in the MMC specification, is not supported.

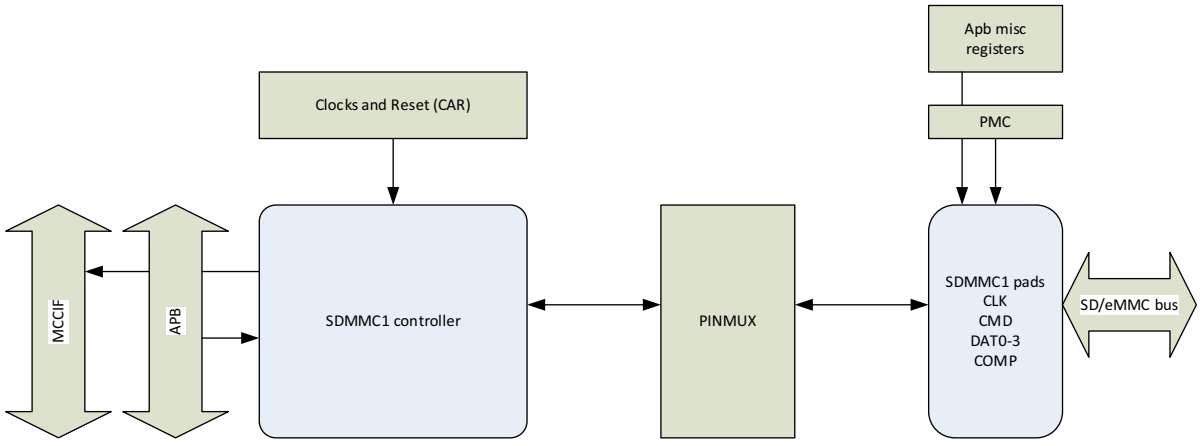
32.5 SD/MMC Interfaces

Table 201: SD Interfaces on Tegra X1

Controller	Number of Pinmuxings		Data Bus width	Pads Used	I/O Voltage (V)	Max. I/O Clock supported (MHz)	Speed modes Supported			Typical Use Case	Max. BW (MBps)
SDMMC1	1	SDMMC1	4	BDSDMEM_B	3.3/1.8	208	DS, HS, SDR12, SDR25, SDR50 and SDR104	DS, HS, SDR12, SDR25, SDR50 and SDR104	LEGACY SPEED, HIGH SPEED SDR, HIGH SPEED DDR and HS200 in 4-bit mode	SDIO for WiFi or removable SD card	104
SDMMC2	1	SDMMC2	8	BDEMMC _IOBRICK_BFC	1.8	208 Or 333	Removable SD card not supported	DS, HS, SDR12, SDR25, SDR50 and SDR104	LEGACY SPEED, HIGH SPEED SDR, HIGH SPEED DDR, HS200, HS400 and HS533	SDIO for secondary Modem Or eMMC5.0 device	667
SDMMC3	1	SDMMC3	4	BDSDMEM_B	3.3/1.8	208	DS, HS, SDR12, SDR25, SDR50 and SDR104	DS, HS, SDR12, SDR25, SDR50 and SDR104	LEGACY SPEED, HIGH SPEED SDR, HIGH SPEED DDR and HS200 in 4-bit mode	Removable SD card or SDIO device	104
SDMMC4	1	SDMMC4	8	BDEMMC _IOBRICK_BFC	1.8	333	Removable SD card not supported	DS, HS, SDR12, SDR25, SDR50 and SDR104	LEGACY SPEED, HIGH SPEED SDR, HIGH SPEED DDR, HS200, HS400 and HS533	Bootable eMMC	667

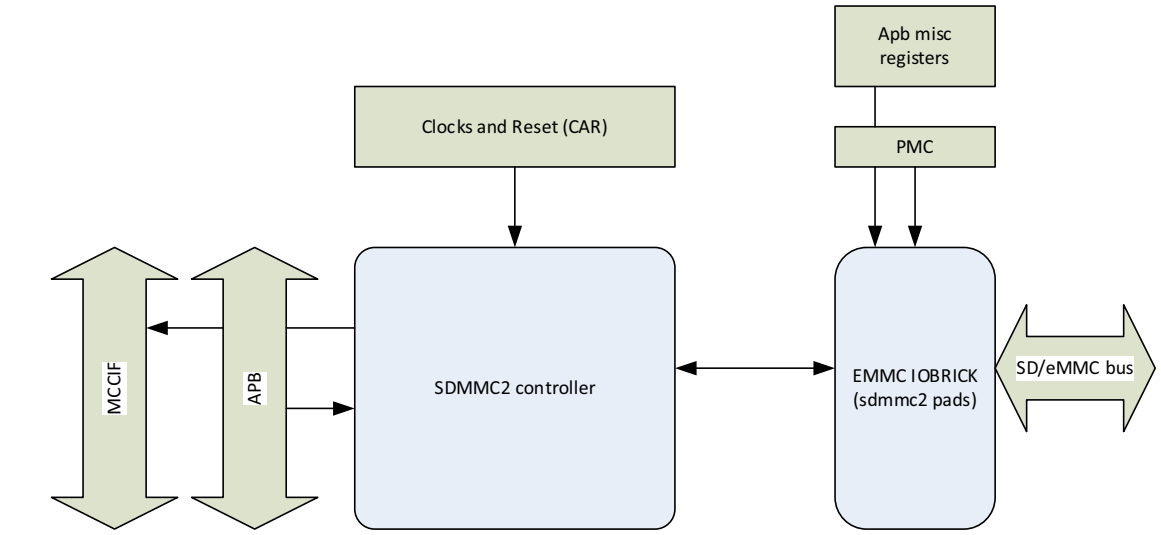
The diagrams below show how the SDMMC controller is hooked up in the SoC.

Figure 148: SDMMC 1 Interface



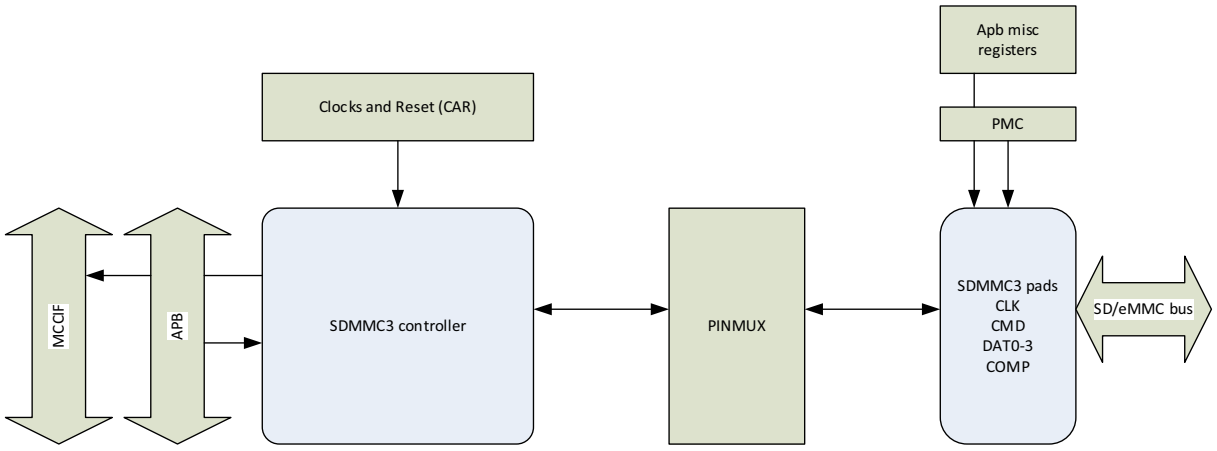
*COMP is compensation pad used for pad drive strength calibration.

Figure 149: SDMMC 2 Interface



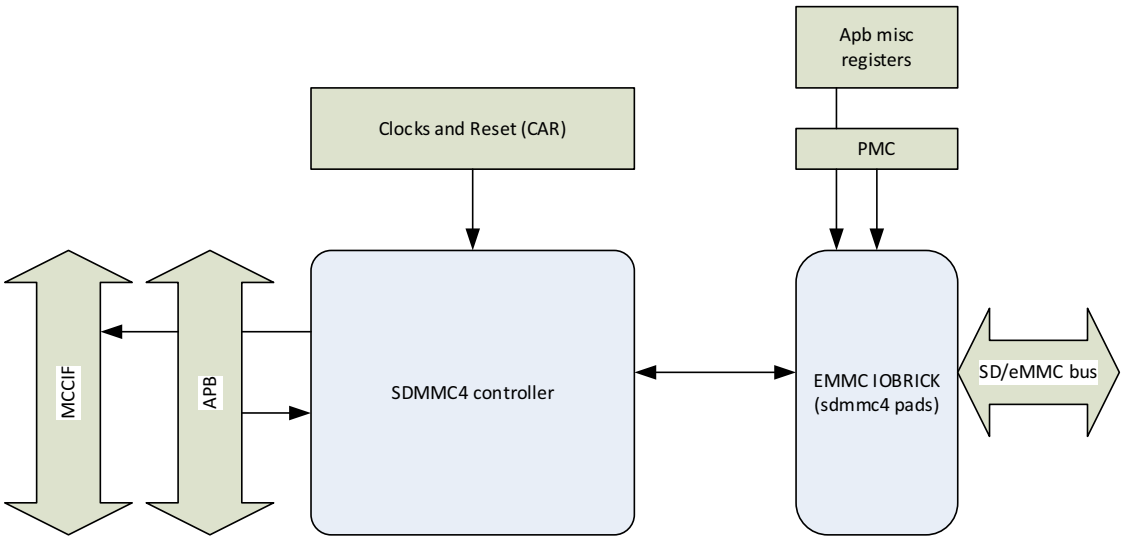
*COMP is compensation pad used for pad drive strength calibration.

Figure 150: SDMMC 3 Interface



*COMP is compensation pad used for pad drive strength calibration.

Figure 151: SDMMC 4 Interface



*COMP is compensation pad used for pad drive strength calibration.

32.6 Pinmux Options

The table below provides an overview of the Pinmux configuration for SDMMC controllers. Software should use the Pinmux spreadsheet provided along with the TRM to get full details.

Table 202: SD/MMC Pinmux Options

Controller	Config	Signal	Ball Name	PINGROUP
SDMMC1	sdmmc1_cmd_cfg	SDMMC1_CMD	SDMMC1_CMD	sdmmc1_cmd_pm
	sdmmc1_clk_cfg	SDMMC1_CLK	SDMMC1_CLK	sdmmc1_clk_pm
	sdmmc1_dat0_cfg	SDMMC1_DAT0	SDMMC1_DAT0	sdmmc1_dat0_pm
	sdmmc1_dat1_cfg	SDMMC1_DAT1	SDMMC1_DAT1	sdmmc1_dat1_pm
	sdmmc1_dat2_cfg	SDMMC1_DAT2	SDMMC1_DAT2	sdmmc1_dat2_pm
	sdmmc1_dat3_cfg	SDMMC1_DAT3	SDMMC1_DAT3	sdmmc1_dat3_pm
	gpio_pz4_cfg	SDMMC1_WP	GPIO_PZ4	gpio_pz4_pm
	gpio_pz1_cfg	SDMMC1_CD	GPIO_PZ1	gpio_pz1_pm
SDMMC2	emmccfg2	SDMMC2_CMD	SDMMC2_CMD	IOBRICK
	emmccfg2	SDMMC2_CLK	SDMMC2_CLK	IOBRICK
	emmccfg2	SDMMC2_CLKB	SDMMC2_CLKB	IOBRICK
	emmccfg2	SDMMC2_DAT0	SDMMC2_DAT0	IOBRICK
	emmccfg2	SDMMC2_DAT1	SDMMC2_DAT1	IOBRICK
	emmccfg2	SDMMC2_DAT2	SDMMC2_DAT2	IOBRICK
	emmccfg2	SDMMC2_DAT3	SDMMC2_DAT3	IOBRICK
	emmccfg2	SDMMC2_DAT4	SDMMC2_DAT4	IOBRICK
	emmccfg2	SDMMC2_DAT5	SDMMC2_DAT5	IOBRICK
	emmccfg2	SDMMC2_DAT6	SDMMC2_DAT6	IOBRICK
	emmccfg2	SDMMC2_DAT7	SDMMC2_DAT7	IOBRICK
	emmccfg2	SDMMC2_DQS	SDMMC2_DQS	IOBRICK
	emmccfg2	SDMMC2_DQSB	SDMMC2_DQSB	IOBRICK
SDMMC3	sdmmc3_cmd_cfg	SDMMC3_CMD	SDMMC3_CMD	sdmmc3_cmd_pm
	sdmmc3_clk_cfg	SDMMC3_CLK	SDMMC3_CLK	sdmmc3_clk_pm
	sdmmc3_dat0_cfg	SDMMC3_DAT0	SDMMC3_DAT0	sdmmc3_dat0_pm
	sdmmc3_dat1_cfg	SDMMC3_DAT1	SDMMC3_DAT1	sdmmc3_dat1_pm
	sdmmc3_dat2_cfg	SDMMC3_DAT2	SDMMC3_DAT2	sdmmc3_dat2_pm
	sdmmc3_dat3_cfg	SDMMC3_DAT3	SDMMC3_DAT3	sdmmc3_dat3_pm
	gpio_pz3_cfg	SDMMC3_WP	GPIO_PZ3	gpio_pz3_pm
	gpio_pz2_cfg	SDMMC3_CD	GPIO_PZ2	gpio_pz2_pm
SDMMC4	emmccfg4	SDMMC4_CMD	SDMMC4_CMD	IOBRICK
	emmccfg4	SDMMC4_CLK	SDMMC4_CLK	IOBRICK
	emmccfg4	SDMMC4_CLKB	SDMMC4_CLKB	IOBRICK
	emmccfg4	SDMMC4_DAT0	SDMMC4_DAT0	IOBRICK
	emmccfg4	SDMMC4_DAT1	SDMMC4_DAT1	IOBRICK
	emmccfg4	SDMMC4_DAT2	SDMMC4_DAT2	IOBRICK
	emmccfg4	SDMMC4_DAT3	SDMMC4_DAT3	IOBRICK
	emmccfg4	SDMMC4_DAT4	SDMMC4_DAT4	IOBRICK
	emmccfg4	SDMMC4_DAT5	SDMMC4_DAT5	IOBRICK
	emmccfg4	SDMMC4_DAT6	SDMMC4_DAT6	IOBRICK
	emmccfg4	SDMMC4_DAT7	SDMMC4_DAT7	IOBRICK
	emmccfg4	SDMMC4_DQS	SDMMC4_DQS	IOBRICK
	emmccfg4	SDMMC4_DQSB	SDMMC4_DQSB	IOBRICK

32.6.1 SD CD# and SDWP# Pins

The removable SD cards' Card Detect and Write protect pins apply to SDMMC1 and SDMMC3 only.

32.7 Programming Guidelines

32.7.1 Introduction

This section assumes a SD Host driver complying with the SD Specifications' Part A2, SD Host Control Standard Specification 4.0, is already available. The following subsections detail the necessary changes required for fully-featured SD (and eMMC boot mode) operation.

32.7.2 Initialization

As a part of initialization, software would be enabling pads, pinmux, SDMMC controller clocks by programming PLLs and CLK dividers, programming Host controller vendor registers, device initialization and tuning. Once initialization is done, SDMMC controller will be ready for executing data transfers.

Each SDMMC controller is functionally the same but differ in data bus supported and speed modes supported. And also each controller has separate register set for programming CLK sources, pad controls and PINMUX settings. Full details of each and every step involved in this initialization process per controller is given in the following sections.

32.7.2.1 SDMMC1 Initialization Sequence

This section provides SDMMC1 controller initialization sequence which should be followed by software driver before doing any CMD/DAT transfers to external SD/SDIO device.

Power ON

1. Make sure SDMMC1 controller is powered up by programming external PMU/PMIC (platform specific) to set VDD_CORE voltage.
2. Refer to the PMU/PMIC datasheet and power up sequence section in this TRM for programming details.

Program Clocks and Reset

Refer to Chapter 5: Clock and Reset Controller for full details of PLL programming used for SDMMC1 clocking.

3. Set SDMMC1 Reset:
   a. Keep SDMMC1 in reset by writing

      CLK_RST_CONTROLLER_RST_DEV_L_SET_0_ SET_SDMMC1_RST with ENABLE (0x1).
4. Program CLK divider and source
   a. Program SDMMC1 host clock divider

      (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0_ SDMMC1_CLK_DIVISOR) and PLL source (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0_ SDMMC1_CLK_SRC) as per below guidelines.
   b. PLLP and PLLC4 are the main CLK sources for SDMMC1 controller.
      - PLLP: VCO running at fixed 408 MHz
      - PLLC4: VCO running at fixed ~1GHz (998.4MHz +/-18ppm quality) - Dedicated PLL for eMMC5.0 HS400 and HS667 clocking. This can also be used for clocking other lower speed modes for SD/SDIO.
      - For SD/SDIO interfaces, it is preferable to source from PLLP for power saving.
      - PLLC4 clock jitter and DCD (Duty Cycle Distortion) are optimized for HS400/533 usage. PLLP can't be used for HS400/533. In other modes, software can use PLLP always.
   c. Software driver should program SDCLK Frequency Select in the Clock Control Register to get the desired SD/SDIO device clock which is derived from SDMMC host clock. Follow SD clock supply sequence mentioned in SD Host specification 4.0.

d. Maximum operating host clock frequency for SDMMC1 controller is 208 MHz.

e. Software may choose to run SDMMC1 controller host clock at a lower speed also.

f. The initializing frequency up to 400 KHz is provided to SD/SDIO card using SDCLK Frequency Select (current maximum divisor 2046 as per SD host specification). This is completely software dependent.

g. In HIGH SPEED DDR/DDR50 modes, SDMMC1 host clock should be run at 2X frequency than device clock. This requires SDMMC internal clock divider should be set to '2' [Standard Host Register SDMMCA_SW_RESET_TIMEOUT_CTRL_CLOCK_CONTROL_0_ SDCLK_FREQUENCYSELECT value would be '1']. This is mandatory to make DDR50/52 mode work.

h. Note there is no SDCLK Frequency Select restriction in SDR modes (DS, HS, SDR12, SDR25, SDR50, SDR104 and HS200).

The following table summarizes the CLK divider options for the PLLP clock source.

Table 203: PLL CLK Divider Options for PLLP

Speed Mode	Target I/O Max Frequency (MHz)	Clock Source Name[1]	Clock Source Frequency (MHz)	CAR Divider[2]	SDMMC Host Clock Frequency (MHz)	SDMMC Divider[3]	Actual I/O Frequency (MHz)
Identification	400 (kHz)	PLLP_OUT0	408	16.5	24.72727273	62	398.83 (kHz)
Default Speed	25	PLLP_OUT0	408	16.5	24.72727273	1	24.72727273
High Speed	50	PLLP_OUT0	408	8.5	48	1	48
SDR12	25	PLLP_OUT0	408	16.5	24.72727273	1	24.72727273
SDR25	50	PLLP_OUT0	408	8.5	48	1	48
SDR50	100	PLLP_OUT0	408	4.5	90.66666667	1	90.66666667
DDR50	50	PLLP_OUT0	408	5	81.6	2	40.8
SDR104	208	PLLP_OUT0	408	2	204	1	204

1. It is set by SDMMC1_CLK_SRC in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0.
2. It is set by SDMMC1_CLK_DIVISOR in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0.
3. It is set by SDCLK Frequency Select in Clock Control Register.

5. Program SDMMC1 timeout clock source and divider

   a. SDMMC1 uses 12MHz TMCLK which is advertised in Host capabilities register -SDMMCA_CAPABILITIES_0_ TIMEOUT_CLOCK_FREQUENCY.

   b. Software should program below registers to provide TMCLK to SDMMC1.

      - CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_ SDMMC_LEGACY_TM_CLK_SRC
      - CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_ SDMMC_LEGACY_TM_CLK_DIVISOR

   c. PLLP_OUT0 is the recommended TMCLK source for SDMMC1.

6. Enable SDMMC1 host clock and TMCLK by setting below registers.

   a. Enable Host clock for SDMMC1 by writing 1 into SET_CLK_ENB_SDMMC1 field of register CLK_RST_CONTROLLER_CLK_ENB_L_SET_0.

   b. Enable TMCLK for SDMMC1 by writing 1 into CLK_ENB_SDMMC_LEGACY_TM_ENABLE field of register CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.

7. Wait for at least 100 sdmmc1 host clock cycles time to get reset propagated down the pipe and also to get internal clocks stabilized.

8. Clear SDMMC1 reset by writing CLK_RST_CONTROLLER_RST_DEV_L_CLR_0_ CLR_SDMMC1_RST with ENABLE. Enable takes effect immediately.

Card Detection and Write Protect Status

9. Tegra X1 supports GPIO based card detection mechanism. Refer to Chapter 9: Multi-Purpose I/O Pins and Pin Multiplexing (Pinmuxing) in this TRM for full details about GPIO implementation.
    a. Software should program SDMMC1_CD and SDMMC1_WP pinmux registers as per customer pinmux datasheet.
10. SDMMC1 controller gets SDCD and SDWP status from GPIO pins (SDMMC1_CD and SDMMC1_WP). Software has to program CD/WP source appropriately in below register to get SDMMC1 correct status.
    a. APB_MISC_GP_VGPIO_GPIO_MUX_SEL_0_ SDMMC1_WP_SOURCE = GPIO
    b. APB_MISC_GP_VGPIO_GPIO_MUX_SEL_0_ SDMMC1_CD_SOURCE = GPIO

Power ON SD card or SDIO device

11. If removable card is detected using GPIO, power ON SD card (3.3V to start with). No need to wait for card detection in case of SDIO.
12. To avoid high leakage and pad damage, set

    SDMMCA_SDMEMCOMPPADCTRL_0_ PAD_E_INPUT_OR_E_PWRD to 0x1 before turning ON SD card power.
13. Program PWR_DET for SDMMC1 pads before turning ON SD/SDIO card power:
    a. SDMMC1 pads are capable of supporting both 3.3V and 1.8V I/O. But software needs to notify pads about the operating I/O voltage whether it is either 3.3V or 1.8V. This is needed as SDMMC1 pads don't have PWR_DET cells to detect voltage level automatically. Incorrect programming could lead to pad damage.
    b. Set APBDEV_PMC_PWR_DET_VAL_0_SDMMC1 = 1 before switching I/O voltage to 3.3V
    c. Set APBDEV_PMC_PWR_DET_VAL_0_SDMMC1 = 0 after switching I/O voltage to 1.8V

PINMUX and Pad Control Programming

SDMMC1 pads are pinmuxed. Software has to program PINMUX registers to select SDMMC1 and also to program pad controls.

14. Program CLK pad pinmux and control settings:
    a. Select 33 Ohm driver - PINMUX_AUX_SDMMC1_CLK_0_ DRV_TYPE_ DRIVE_2X. Ensure that the SDMMC1_COMP pad is connected to GND via 66 Ohm resistor on board.
    b. DRV_TYPE_DRIVE2X and 66 Ohm calibration resistor are used to get drive strength codes for 33 ohm driver. This is Tegra specific implementation.
    c. Schmitt trigger selection
    - PINMUX_AUX_SDMMC1_CLK_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock
    - PINMUX_AUX_SDMMC1_CLK_0_ E_SCHMT_DISABLE for 3.3V I/O
    d. Both E_INPUT and E_LPBK of SDMMC1 CLK pad should be enabled to provide loopback CLK for SDMMC1 Receiver. If disabled, reads from SD/eMMC device won't work.
    - APB_MISC_GP_SDMMC1_CLK_LPBK_CONTROL_0_SDMMC1_CLK_PAD_E_LPBK = 1'b1
    - PINMUX_AUX_SDMMC1_CLK_0_E_INPUT = ENABLE
    e. Remove tri-state to enable pad output driver - PINMUX_AUX_SDMMC1_CLK_0_TRISTATE_ PASSTHROUGH
    f. Select SDMMC1 in pinmux - PINMUX_AUX_SDMMC1_CLK_0_PM_SDMMC1
    g. Other pad control settings in PINMUX_AUX_SDMMC1_CLK_0 register should be left as default values. No need to modify.
15. Program CMD pad pinmux and control settings:
    a. Select 33 Ohm driver - PINMUX_AUX_SDMMC1_CMD_0_ DRV_TYPE_ DRIVE_2X
    b. Schmidt trigger selection
    - PINMUX_AUX_SDMMC1_CMD_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock

- PINMUX_AUX_SDMMC1_CMD_0_ E_SCHMT_DISABLE for 3.3V I/O

c. E_INPUT of SDMMC1 CMD pad should be enabled to activate receiver in CMD pad. If disabled, response sent by SD/SDIO will be lost.

- PINMUX_AUX_SDMMC1_CMD_0_E_INPUT_ENABLE

d. Remove tri-state to enable pad output driver - PINMUX_AUX_SDMMC1_CMD_0_TRISTATE_ PASSTHROUGH

e. Select SDMMC1 in pinmux - PINMUX_AUX_SDMMC1_CMD_0_PM_SDMMC1

f. Other pad control settings in PINMUX_AUX_SDMMC1_CMD_0 register should be left as default values. No need to modify.

16. Program DAT0/1/2/3 pad pinmux and control settings:

a. Select 33 Ohm driver - PINMUX_AUX_SDMMC1_DATx_0_ DRV_TYPE_ DRIVE_2X

b. Schmitt trigger selection

- PINMUX_AUX_SDMMC1_DATx_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock

- PINMUX_AUX_SDMMC1_DATx_0_ E_SCHMT_DISABLE for 3.3V I/O

c. E_INPUT of SDMMC1 DATx pad should be enabled to activate receiver in DATx pad. If disabled, data sent by SD/SDIO will be lost.

- PINMUX_AUX_SDMMC1_DATx_0_E_INPUT_ENABLE

d. Remove tri-state to enable pad output driver - PINMUX_AUX_SDMMC1_DATx_0_TRISTATE_ PASSTHROUGH

e. Select SDMMC1 in pinmux - PINMUX_AUX_SDMMC1_DATx_0_PM_SDMMC1

f. Other pad control settings in PINMUX_AUX_SDMMC1_DATx_0 register should be left as default values. No need to modify.

Note: *x = 0, 1, 2 or 3 – DAT lane number*

Program Pad Control and Vendor Registers

17. Program below registers before running auto-calibration to prevent CRC errors during data transfers.

a. Write 0x1 into bit 19 (SPARE_OUT[3]) of register SDMMCA_IO_SPARE_0.

b. Write 0x0 into bit 2 (SEL_VREG) of register SDMMCA_VENDOR_IO_TRIM_CNTRL_0.

Refer to Section 32.8.3: DLL Calibration for software sequence to be followed to program this register field.

18. Set outbound clock trimmer tap value (works in all speed modes - up to 208 MHz) before doing any cmd/data transfers to device. This is needed to meet interface timing specification.

a. SDMMCA_VENDOR_CLOCK_CNTRL_0_ TRIM_VAL = 0x2

19. Set inbound clock trimmer tap value (works up to 50MHz – in non-tunable modes such as DS, HS, SDR12, SDR25 and DDR50) before doing any cmd/data transfers to device. Tuning procedure needs to be run in other modes – SDR50 and SDR104.

a. SDMMCA_VENDOR_CLOCK_CNTRL_0_ TAP_VAL = 0x4

20. Set CLK pad slew codes to 0x0 to get good quality clock.

a. APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_ CFG2TMC_SDMMC1_CLK_CFG_CAL_DRVUP_SLWF = 0x1

b. APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_ CFG2TMC_SDMMC1_CLK_CFG_CAL_DRVDN_SLWR = 0x1

21. Set Calibration pad VSEL:

a. SDMMCA_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7

Run Auto-Calibration

Run auto-calibration procedure to set proper drive strength codes in SDMMC1 pads. This is needed to use 33 Ohm driver for driving SD/SDIO interface. This procedure should be run before doing device initialization when SD card I/O is using 3.3V and SDIO is using 1.8V.

Software also needs to re-run auto-calibration immediately after switching to 1.8V I/O from 3.3V I/O and before doing any transfers to SD device.

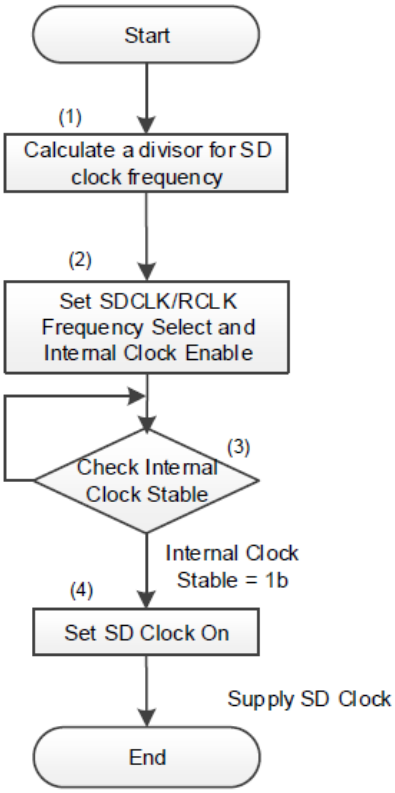
22. Program drive code offsets when for 3.3V operation before running auto-calibration:
    a. SDMMCA_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8'd125
    b. SDMMCA_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8'd0
23. Program drive code offsets for 1.8V operation before running auto-calibration:
    a. SDMMCA_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8'd123
    b. SDMMCA_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8'd123
24. Run auto-calibration procedure. Ensure that the SD/eMMC device clock is off during the calibration procedure.
    a. Set E_INPUT_OR_E_PWRD = 1 in SDMMCA_SDMEMCOMPPADCTRL_0 register, if not set.
    b. Wait for 1 us after E_INPUT_OR_E_PWRD is enabled.
    c. Set AUTO_CAL_START and AUTO_CAL_ENABLE to 1 in SDMMCA_AUTO_CAL_CONFIG_0 register.
    d. Wait for 1 us
    e. Wait for up to 10 ms for AUTO_CAL_ACTIVE in SDMMCA_AUTO_CAL_STATUS_0 register to become 0. If it is not 0 after 10 ms, software can assume auto-calibration has timed out. If it becomes zero within 10ms, it is assumed that calibration has completed.
    f. Clear E_INPUT_OR_E_PWRD to save power.
    g. Software should not clear AUTO_CAL_ENABLE in SDMMCA_AUTO_CAL_CONFIG_0 register after calibration. It should be set to 1 always to use calibration codes generated by calibration controller.
    h. If timeout occurs in auto-calibration process, Software should program the drive strength code values below. Software should clear AUTO_CAL_ENABLE in SDMMCA_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.
    i. For 3.3V I/O:
        - APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC1_PAD_CAL_DRVUP = 0xC
        - APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC1_PAD_CAL_DRVDN = 0xC
    j. For 1.8V I/O:
        - APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC1_PAD_CAL_DRVUP = 0xB
        - APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC1_PAD_CAL_DRVDN = 0xF
    k. Enable periodic calibration to compensate for drift in driver strength due to temperature variation.
        - Re-run calibration for every 100 ms.
        - Start 100 ms timer after completing calibration iteration.
        - Once 100 ms timer expires, software has to re-initiate calibration as per above sequence but software needs to check if there are any ongoing transfers or not.

          If there are no transfers, calibration can be triggered immediately or before a new data transfer starts. If a new data transfer request comes from upper OS layers during calibration, software should stall that request till calibration is completed.

          If there are ongoing transfers, software should wait for transfers to complete and re-initiate calibration.

25. Enable SD device clock by writing into SD Host registers (should follow SD bus power and Clock supply sequence described in section 3.2 of SD Host 4.0 specification). The SD bus power field in Power Control register should be set to 1 before enabling SDCLK. A flowchart of the SD clock supply sequence is given for quick reference below.

Figure 152: SD Clock Supply Sequence



26. Do SD/SDIO device initialization as per the sequence mentioned in Section 3.6 of SD Host 4.0 specification published by SD Card Association.
27. Points to remember while doing device initialization:
    a. During device initialization, when I/O power is switched to 1.8V from 3.3V, software needs to run auto-calibration again to set proper drive strengths before accessing external device.
    b. During voltage switching, follow the guidelines below to avoid pad damage.
        - Make sure APBDEV_PMC_PWR_DET_VAL_0_SDMMC1 = 1 when I/O voltage is 3.3V.
        - Program PMIC to switch I/O voltage to 1.8V from 3.3V.
        - Wait for 1 ms to cover LDO ramp time from 3.3V to 1.8V (5 mV/us)
        - Set APBDEV_PMC_PWR_DET_VAL_0_SDMMC1 = 0 since I/O voltage is switched to 1.8V.
        - Note I/O power switching can be done by programming PMU/PMIC along with programming the SD Host Power Control register, 1.8V signaling enable in Host Control2 register.
        - Program vendor registers and pad controls for 1.8V I/O as mentioned in previous sections, if needed.
28. Software driver needs to set inbound clock trimmer tap value as per below guidelines to fix sampling point for data coming from device.
    a. In non-tunable modes (DS, HS, SDR12, SDR25 and DDR50 – up to 50 MHz), Software needs to program inbound trimmer tap value as mentioned in Inbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL).
    b. To do data transfers in tunable modes (SDR50 and SDR104), the tuning procedure needs to be run to fix sampling point first. For Tuning procedure details, refer to Section 32.8.9: SDR50/SDR104/HS200 Tuning Procedure.
29. Now, Host and device are ready for doing a data transfers.

30. Follow software programming sequences mentioned in Chapter 3 of SD Host 4.0 specification published by SD Card Association for doing data/cmd transfers.

32.7.2.2 SDMMC2 Initialization Sequence

This section provides SDMMC2 controller initialization sequence which should be followed by the software driver before doing any CMD/DAT transfers to external eMMC device.

Power ON

1. Make sure SDMMC2 controller is powered up by programming external PMU/PMIC (platform specific) to set VDD_CORE voltage.
2. Refer to the PMU/PMIC datasheet and power up sequence section in this TRM for programming details.

Program Clocks and Reset

Refer to Chapter 5: Clock and Reset Controller for full details of PLLs programming which are used for SDMMC2 clocking.

3. Set SDMMC2 reset:
   a. Keep SDMMC2 in reset by writing CLK_RST_CONTROLLER_RST_DEV_L_SET_0_ SET_SDMMC2_RST with ENABLE (0x1).
4. Program CLK divider and source
   a. Program SDMMC2 host clock divider

      (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0_ SDMMC2_CLK_DIVISOR) and PLL source

      (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0_ SDMMC2_CLK_SRC) as per below guidelines.
   b. PLLP and PLLC4 are the main CLK sources for SDMMC2 controller.
      - PLLP: VCO running at fixed 408 MHz
      - PLLC2: VCO running at fixed ~1GHz (998.4 MHz +-18ppm quality) - Dedicated PLL for eMMC5.0 HS400 and HS667 clocking.
      - For eMMC HS400/HS533 modes, software should use PLLC4 only since PLLC4 clock path is optimized to meet HS400 duty cycle distortion specification requirements.
   c. Software driver should program SDCLK Frequency Select to get desired SD/SDIO device clock which is derived from SDMMC host clock. Follow the SD clock supply sequence mentioned in the SD Host specification 4.0.
   d. Maximum operating host clock frequency for SDMMC2 controller is 333 MHz.
   e. The initializing frequency up to 400 kHz is provided to eMMC using the SDCLK Frequency Select of SDMMC2 (current maximum divisor 2046 as per SD host specification). This is completely software dependent.
   f. In HIGH SPEED DDR mode, SDMMC2 host should be run at 2X frequency than I/O. This requires SDMMC internal clock divider should be set to ‘2’ [Standard Host Register SDMMCAA_SW_RESET_TIMEOUT_CTRL_CLOCK_CONTROL_0_ SDCLK_FREQUENCYSELECT value would be ‘1’].This is mandatory to make DDR50/52 mode work.
   g. Note there is no SDCLK Frequency Select restriction in SDR modes (DS, HS, SDR12, SDR25, SDR50, SDR104 and HS200).
   h. HS400/HS533 mode: This is DDR mode only but has different timing and protocol compared to DDR50/52 mode. In this mode, host clock frequency should match I/O clock frequency. This is a Tegra design requirement. There is no specification compliance issue due to this as this mode is specific to eMMC devices and not defined in SD host specification.

The following table summarizes CLK divider options for PLLP, PLLC4 clock sources.

Table 204: PLL CLK Divider Options for PLLP and PLLC4

Speed Mode	Target I/O Max Frequency (MHz)	Clock Source[1]	Clock Source Frequency (MHz)	CAR Divider[2]	SDMMC host Clock Frequency (MHz)	SDMMC Divider[3]	Actual I/O Frequency (MHz)
Identification	400 (kHz)	PLLP_OUT0	408	16.5	24.72727273	62	398.83 (kHz)
LEGACY SPEED	26	PLLP_OUT0	408	16	25.5	1	25.5
HIGH SPEED SDR	52	PLLP_OUT0	408	8	51	1	51
HIGH SPEED DDR	52	PLLP_OUT0	408	4	102	2	51
HS200 (SDR)	200	PLLC4_OUT2_LJ	199.68	1	199.68	1	199.68
HS400 (DDR)	200	PLLC4_OUT2_LJ	199.68	1	199.68	1	199.68
HS533 (DDR)	266	PLLC4_OUT0_LJ (DIVP=4)	249.6	1	249.6	1	249.6

1. It is set by SDMMC2_CLK_SRC in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0.
2. It is set by SDMMC2_CLK_DIVISOR in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0.
3. It is set by SDCLK Frequency Select in Clock Control Register.

5. Program SDMMC2 timeout clock source and divider
   a. SDMMC2 uses 12 MHz TMCLK which is advertised in Host capabilities register -SDMMCAA_CAPABILITIES_0_ TIMEOUT_CLOCK_FREQUENCY.
   b. Software should program below registers to provide TMCLK to SDMMC2.
   - CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_ SDMMC_LEGACY_TM_CLK_SRC
   - CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_SDMMC_LEGACY_TM_CLK_DIVISOR
   c. PLLP_OUT0 is the recommended TMCLK source for SDMMC2.
6. Enable SDMMC2 host clock and TMCLK by setting below registers.
   a. Enable SET_CLK_ENB_SDMMC2 in CLK_RST_CONTROLLER_CLK_ENB_L_SET_0.
   b. Enable CLK_ENB_SDMMC_LEGACY_TM in CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.
7. Wait for at least 100 SDMMC2 host clock cycles time to get reset propagated down the pipe and also to get internal clocks stabilized.
8. Clear SDMMC2 reset by writing CLK_RST_CONTROLLER_RST_DEV_L_CLR_0_ CLR_SDMMC2_RST with ENABLE. Enable takes effect immediately.

Power ON eMMC or SDIO Device

9. Power ON eMMC (1.8V I/O). No need to wait for card detection as it is an embedded device.

PINMUX and eMMC IOBRICK Control Programming

10. SDMMC2 uses IOBRICK which is not pinmuxed. No PINMUX programming is needed.
11. E_INPUT and E_LPBK for SDMMC2 CLK pin should be enabled to provide loopback CLK for SDMMC2 Rx. If disabled, reads from eMMC device will not work.
    a. APB_MISC_GP_EMMC2_PAD_CFG_CONTROL_0_EMMC2_PAD_E_DEEP_LPBK_CLK = 1’b1
    b. APB_MISC_GP_EMMC2_PAD_CFG_CONTROL_0_ EMMC2_PAD_E_INPUT_CLK = 1’b1
12. Program below register fields in APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0 register.
    a. APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC2_PAD_DRVUP_COMP = 0x10
    b. APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC2_PAD_DRVDN_COMP = 0x10
    c. The other bits in register APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0 must not be touched by software. Default values should be used.
13. Select 50 Ohm driver:

a. Ensure SDMMC2_COMP pad is connected to GND via 50 Ohm resistor on board.

b. Make sure all DRV_TYPE fields in APB_MISC_GP_EMMC2_PAD_DRV_TYPE_CFGPADCTRL_0 are not updated by software. Software should use default values (DRV_TYPE_2X) only.

14. Pull Up/Pull Down values in APB_MISC_GP_EMMC2_PAD_PUPD_CFGPADCTRL_0 should match with default values. No updates are needed.

Program Host Vendor Registers

15. Program below registers before running auto-calibration to prevent CRC errors during data transfers done later.

    a. Write 0x1 into bit 19 (SPARE_OUT[3]) of register SDMMCAA_IO_SPARE_0.

    b. Write 0x0 into bit 2 (SEL_VREG) of register SDMMCAA_VENDOR_IO_TRIM_CNTRL_0. Refer to Section 32.8.5: DLL and Inbound Clock Trimmer Power Supply (VREG) Programming for the software sequence to be followed to program this register field.

16. Set outbound clock trimmer tap value (works in all speed modes - up to 200 MHz) before doing any cmd/data transfers to device. This is needed to meet interface timing specification.

    a. SDMMCAA_VENDOR_CLOCK_CNTRL_0_ TRIM_VAL = 0x8

17. Set internal DQS trimmer tap value before doing any cmd/data transfers to device in HS400/533 mode. This is needed to meet interface timing specification.

    a. For device clock frequency <=200 MHz:

       SDMMCAA_VENDOR_CAP_OVERRIDES_0_ DQS_TRIM_VAL = 40 (decimal)

    b. For device clock frequency > 200MHz and <=333 MHz:

       SDMMCAA_VENDOR_CAP_OVERRIDES_0_ DQS_TRIM_VAL = 24 (decimal)

18. Set inbound clock trimmer tap value (works up to 50 MHz – in non-tunable modes such as SDR@26, SDR@52 and HIGH SPEED DDR) before doing any cmd/data transfers to device. Tuning procedure needs to be run in other modes – HS200 and HS400/533.

    a. SDMMCAA_VENDOR_CLOCK_CNTRL_0_ TAP_VAL = 0x0

19. Set Calibration pad VSEL:
    SDMMCAA_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7

Run Auto-Calibration

Run auto-calibration procedure to set proper drive strength codes for SDMMC2 IOBRICK. This is needed to use 50 Ohm driver for driving eMMC interface. This procedure should be run before doing device initialization.

20. Program drive code offsets for 1.8V operation before running auto-calibration:

    a. SDMMCAA_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8’d5

    b. SDMMCAA_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8’d5

21. Run auto-calibration procedure. Ensure that the SD/eMMC device clock is off during calibration procedure.

    a. Set E_INPUT_OR_E_PWRD = 1 in SDMMCAA_SDMEMCOMPPADCTRL_0 register, if not set.

    b. Wait for 1us after E_INPUT_OR_E_PWRD is enabled.

    c. Set AUTO_CAL_START and AUTO_CAL_ENABLE to 1 in SDMMCAA_AUTO_CAL_CONFIG_0 register.

    d. Wait for 1 us.

    e. Wait for up to 10 ms for AUTO_CAL_ACTIVE in SDMMCAA_AUTO_CAL_STATUS_0 register to become 0. If it is not 0 after 10 ms, software can assume auto-calibration has timed out. If it becomes zero within 10 ms, it is assumed that calibration has completed.

    f. Clear E_INPUT_OR_E_PWRD to save power.

    g. Software should not clear AUTO_CAL_ENABLE in SDMMCAA_AUTO_CAL_CONFIG_0 register after calibration. It should be set to 1 always to use calibration codes generated by calibration controller.

h. If timeout occurs in auto-calibration process, software should program below drive strength code values. Software should clear AUTO_CAL_ENABLE in SDMMCAA_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.

i. For 1.8V I/O:

- APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC2_PAD_DRVUP_COMP = 0x10
- APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC2_PAD_DRVDN_COMP = 0x10

22. Enable eMMC device clock by writing into SD Host registers (should follow SD bus power and SD Clock supply sequence described in section 3.2 of SD Host 4.0 specification). See the SD clock supply sequence flowchart in Figure 152. SD bus power field in Power Control register should be set to 1 before enabling SD clock.
23. Do eMMC device initialization as per the sequence mentioned in eMMC5.0 specification published by JEDEC.
24. Run DLL calibration first, if software wants to use HS400/HS533 mode for doing data transfers to/from eMMC. For HS400 mode switching details, refer to Section 32.8.1: HS400 and HS667 Mode Selection.
25. Software driver needs to set inbound clock trimmer tap value as per below guidelines to fix sampling point for data coming from device.
    a. In non-tunable modes (LEGACY SPEED, HIGH SPEED SDR, and HIGH SPEED DDR – up to 52 MHz), software needs to program inbound trimmer tap value as mentioned in Inbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL).
    b. To do data transfers in tunable modes (HS200 and HS400), Tuning procedure needs to be run to fix sampling point first. For Tuning procedure details, refer to Section 32.8.9: SDR50/SDR104/HS200 Tuning Procedure.
26. Now, Host and device are ready for doing a data transfers.
27. Follow software programming sequences mentioned in Chapter 3 of SD Host 4.0 specification published by SDA for doing data/cmd transfers.

32.7.2.3 SDMMC3 Initialization Sequence

This section provides SDMMC3 controller initialization sequence which should be followed by software driver before doing any CMD/DAT transfers to external SD/SDIO device.

Power ON

1. Make sure SDMMC3 controller is powered up by programming external PMU/PMIC (platform specific) to set VDD_CORE voltage.
2. Refer to the PMU/PMIC datasheet and power up sequence section in this TRM for programming details.

Program Clocks and Reset

Refer to Chapter 5: Clock and Reset Controller for full details of PLLs programming which are used for SDMMC3 clocking.

3. Set SDMMC3 reset:
    a. Keep SDMMC3 in reset by writing CLK_RST_CONTROLLER_RST_DEV_U_SET_0_ SET_SDMMC3_RST with ENABLE (0x1).
4. Program CLK divider and source
    a. Program SDMMC3 host clock divider (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0_ SDMMC3_CLK_DIVISOR) and PLL source (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0_ SDMMC3_CLK_SRC) as per below guidelines.
    b. PLLP and PLLC4 are the main CLK sources for SDMMC3 controller.
        - PLLP: VCO running at fixed 408 MHz
        - PLLC4: VCO running at fixed ~1GHz (998.4 MHz +-18ppm quality) - Dedicated PLL for eMMC5.0 HS400 and HS667 clocking. This can also be used for clocking other lower speed modes for SD/SDIO.
        - For SD/SDIO interfaces, it is preferable to source from PLLP for power saving.

c. Software driver should program SDCLK Frequency Select in SD Host Clock control register to get desired SD/SDIO device clock which is derived from SDMMC host clock. Follow SD clock supply sequence mentioned in SD Host specification 4.0.

d. Maximum operating host clock frequency for SDMMC3 controller is 208 MHz.

e. The initializing frequency up to 400 kHz is provided to SD/SDIO card using the internal divider (SDCLK Frequency Select) of SDMMC3 (current maximum divisor 2046 as per SD host specification). This is completely software dependent.

f. In HIGH SPEED DDR/DDR50 modes, SDMMC3 host should be run at 2X frequency than I/O. This requires SDMMC internal clock divider should be set to '2' [Standard Host Register SDMMC_SW_RESET_TIMEOUT_CTRL_CLOCK_CONTROL_0_ SDCLK_FREQUENCYSELECT value would be '1'].This is mandatory to make DDR50/52 mode work.

g. Note there is no SDCLK Frequency Select restriction in SDR modes (DS, HS, SDR12, SDR25, SDR50, SDR104 and HS200).

The following table summarizes the CLK divider options for the PLLP clock source.

Table 205: PLL CLK Divider Options for PLLP Clock Source

Speed Mode	Target I/O Max Frequency (MHz)	Clock Source Name(1)	Clock Source Frequency (MHz)	CAR Divider(2)	SDMMC host Clock Frequency (MHz)	SDMMC divider(3)	Actual I/O Frequency (MHz)
Identification	400 (kHz)	PLLP_OUT0	408	16.5	24.72727273	62	398.83 (kHz)
Default Speed	25	PLLP_OUT0	408	16.5	24.72727273	1	24.72727273
High Speed	50	PLLP_OUT0	408	8.5	48	1	48
SDR12	25	PLLP_OUT0	408	16.5	24.72727273	1	24.72727273
SDR25	50	PLLP_OUT0	408	8.5	48	1	48
SDR50	100	PLLP_OUT0	408	4.5	90.66666667	1	90.66666667
DDR50	50	PLLP_OUT0	408	5	81.6	2	40.8
SDR104	208	PLLP_OUT0	408	2	204	1	204

1. It is set by SDMMC3_CLK_SRC in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0.
2. It is set by SDMMC3_CLK_DIVISOR in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0.
3. It is set by SDCLK Frequency Select in Clock Control Register.

5. Program SDMMC3 timeout clock source and divider

   a. SDMMC3 uses 12 MHz TMCLK which is advertised in Host capabilities register - SDMMC_CAPABILITIES_0_TIMEOUT_CLOCK_FREQUENCY.

   b. Software should program below registers to provide TMCLK to SDMMC3.

      - CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_ SDMMC_LEGACY_TM_CLK_SRC
      - CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_SDMMC_LEGACY_TM_CLK_DIVISOR

   c. PLLP_OUT0 is the recommended TMCLK source for SDMMC3.

6. Enable SDMMC3 host clock and TMCLK by setting below registers.

   a. Enable SET_CLK_ENB_SDMMC3 in CLK_RST_CONTROLLER_CLK_ENB_U_SET_0.

   b. Enable CLK_ENB_SDMMC_LEGACY_TM in CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.

7. Wait for at least 100 SDMMC3 host clock cycles time to get reset propagated down the pipe and also to get internal clocks stabilized.

8. Clear SDMMC3 reset by writing CLK_RST_CONTROLLER_RST_DEV_U_CLR_0_ CLR_SDMMC3_RST with ENABLE. Enable takes effect immediately.

Card Detection and Write Protect Status

9. Tegra X1 supports GPIO based card detection mechanism. Refer to the GPIO section in TRM for full details about GPIO programming.
    a. Software needs to program SDMMC3_CD and SDMMC3_WP pinmux registers as per customer pinmux datasheet.
10. SDMMC3 controller gets SDCD and SDWP status from GPIO pins (SDMMC3_CD and SDMMC3_WP). Software has to program CD/WP source appropriately in below register to get SDMMC3 correct status.
    a. APB_MISC_GP_VGPIO_GPIO_MUX_SEL_0_ SDMMC3_WP_SOURCE = GPIO
    b. APB_MISC_GP_VGPIO_GPIO_MUX_SEL_0_ SDMMC3_CD_SOURCE = GPIO

Power ON SD Card or SDIO Device

11. If removable card is detected using GPIO, power ON SD card (3.3V to start with). No need to wait for card detection in case of SDIO.
12. To avoid high leakage and pad damage, set SDMMC_SDMEMCOMPPADCTRL_0_PAD_E_INPUT_OR_E_PWRD to 0x1 before turning ON SD card power.
13. Program PWR_DET for SDMMC3 pads before turning ON SD/SDIO card power:
    a. SDMMC3 pads are capable of supporting both 3.3V and 1.8V I/O. Software should notify pads about the operating I/O voltage whether it is either 3.3V or 1.8V. This is needed as SDMMC3 pads don't have PWR_DET cells to detect voltage level automatically. Incorrect programming could lead to pad damage.
    b. Set APBDEV_PMC_PWR_DET_VAL_0_SDMMC3 = 1 before switching I/O voltage to 3.3V
    c. Set APBDEV_PMC_PWR_DET_VAL_0_SDMMC3 = 0 after switching I/O voltage to 1.8V

PINMUX and Pad Control Programming

SDMMC3 pads are pinmuxed. Software has to program PINMUX registers to select SDMMC3 and also to program pad controls.

14. Program CLK pad pinmux and control settings:
    a. Select 33 Ohm driver - PINMUX_AUX_SDMMC3_CLK_0_ DRV_TYPE_ DRIVE_2X. Ensure that the SDMMC3_COMP pad is connected to GND via 66 Ohm resistor on board.
    b. DRV_TYPE_DRIVE2X and 66 Ohm calibration resistor are used to get drive strength codes for 33 Ohm driver. This is a Tegra specific implementation.
    c. Schmitt trigger selection
        - PINMUX_AUX_SDMMC3_CLK_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock
        - PINMUX_AUX_SDMMC3_CLK_0_ E_SCHMT_DISABLE for 3.3V I/O
    d. Both E_INPUT and E_LPBK of SDMMC3 CLK pad should be enabled to provide loopback CLK for SDMMC3 Receiver. If disabled, reads from SD/eMMC device won't work.
        - APB_MISC_GP_SDMMC3_CLK_LPBK_CONTROL_0_SDMMC3_CLK_PAD_E_LPBK = 1'b1
        - PINMUX_AUX_SDMMC3_CLK_0_E_INPUT_ENABLE
    e. Remove tri-state to enable pad output driver - PINMUX_AUX_SDMMC3_CLK_0_TRISTATE_PASSTHROUGH
    f. Select SDMMC3 in pinmux - PINMUX_AUX_SDMMC3_CLK_0_PM_SDMMC3
    g. Other pad control settings in PINMUX_AUX_SDMMC3_CLK_0 register should be left as default values. No need to modify.
15. Program CMD pad pinmux and control settings:
    a. Select 33 Ohm driver - PINMUX_AUX_SDMMC3_CMD_0_ DRV_TYPE_ DRIVE_2X
    b. Schmidt trigger selection

- PINMUX_AUX_SDMMC3_CMD_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock
- PINMUX_AUX_SDMMC3_CMD_0_ E_SCHMT_DISABLE for 3.3V I/O

c. E_INPUT of SDMMC3 CMD pad should be enabled to activate receiver in CMD pad. If disabled, response sent by SD/SDIO will be lost.

- PINMUX_AUX_SDMMC3_CMD_0_E_INPUT_ENABLE

d. Remove tri-state to enable pad output driver - PINMUX_AUX_SDMMC3_CMD_0_TRISTATE_ PASSTHROUGH

e. Select SDMMC3 in pinmux - PINMUX_AUX_SDMMC3_CMD_0_PM_SDMMC3

f. Other pad control settings in PINMUX_AUX_SDMMC3_CMD_0 register should be left as default values. No need to modify.

16. Program DAT0/1/2/3 pad pinmux and control settings:

a. Select 33 Ohm driver - PINMUX_AUX_SDMMC3_DATx_0_ DRV_TYPE_ DRIVE_2X

b. Schmitt trigger selection

- PINMUX_AUX_SDMMC3_DATx_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock
- PINMUX_AUX_SDMMC3_DATx_0_ E_SCHMT_DISABLE for 3.3V I/O

c. E_INPUT of SDMMC3 DATx pad should be enabled to activate receiver in DATx pad. If disabled, data sent by SD/SDIO will be lost.

d. PINMUX_AUX_SDMMC3_DATx_0_E_INPUT_ENABLE

e. Remove tri-state to enable pad output driver - PINMUX_AUX_SDMMC3_DATx_0_TRISTATE_ PASSTHROUGH

f. Select SDMMC3 in pinmux - PINMUX_AUX_SDMMC3_DATx_0_PM_SDMMC3

g. Other pad control settings in PINMUX_AUX_SDMMC3_DATx_0 register should be left as default values. No need to modify.

Note: *x = 0, 1, 2 or 3 – DAT lane number.*

Program Pad Control and Vendor Registers

17. Program below registers before running auto-calibration to prevent CRC errors during data transfers done later.

a. Write 0x1 into bit 19 (SPARE_OUT[3]) of register SDMMC_IO_SPARE_0.

b. Write 0x0 into bit 2 (SEL_VREG) of register SDMMC_VENDOR_IO_TRIM_CNTRL_0. Refer to Section 32.8.5: DLL and Inbound Clock Trimmer Power Supply (VREG) Programming for software sequence to be followed to program this register field.

18. Set outbound clock trimmer tap value (works in all speed modes - up to 208 MHz) before doing any cmd/data transfers to device. This is needed to meet the interface timing specification.

a. SDMMC_VENDOR_CLOCK_CNTRL_0_ TRIM_VAL = 0x3

19. Set inbound clock trimmer tap value (works up to 50MHz – in non-tunable modes such as DS, HS, SDR12, SDR25 and DDR50) before doing any cmd/data transfers to device. Tuning procedure needs to be run in other modes – SDR50 and SDR104.

a. SDMMC_VENDOR_CLOCK_CNTRL_0_ TAP_VAL = 0x3

20. Set CLK pad slew codes to 0x0 to get good quality clock.

a. APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_ CFG2TMC_SDMMC3_CLK_CFG_CAL_DRVUP_SLWF = 0x1

b. APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_ CFG2TMC_SDMMC3_CLK_CFG_CAL_DRVDN_SLWR = 0x1

21. Set Calibration pad VSEL: SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7

Run Auto-Calibration

Run auto-calibration procedure to set proper drive strength codes in SDMMC3 pads. This is needed to use 33 Ohm driver for driving SD/SDIO interface. This procedure should be run before doing device initialization when SD card I/O is using 3.3V and SDIO is using 1.8V.

Software also needs to re-run auto-calibration immediately after switching to 1.8V I/O from 3.3V I/O and before doing any transfers to SD device.

22. Program drive code offsets when for 3.3V operation before running auto-calibration:
    a. SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8'd125
    b. SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8'd0
23. Program drive code offsets for 1.8V operation before running auto-calibration:
    a. SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8'd123
    b. SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8'd123
24. Run auto-calibration procedure. Ensure that the SD/eMMC device clock is off during calibration procedure.
    a. Set E_INPUT_OR_E_PWRD = 1 in SDMMC_SDMEMCOMPPADCTRL_0 register, if not set.
    b. Wait for 1us after E_INPUT_OR_E_PWRD is enabled.
    c. Set AUTO_CAL_START and AUTO_CAL_ENABLE to 1 in SDMMC_AUTO_CAL_CONFIG_0 register.
    d. Wait for 1 us.
    e. Wait for up to 10 ms for AUTO_CAL_ACTIVE in SDMMC_AUTO_CAL_STATUS_0 register to become 0. If it is not 0 after 10 ms, software can assume auto-calibration has timed out. If it becomes zero within 10 ms, it is assumed that calibration has completed.
    f. Clear E_INPUT_OR_E_PWRD to save power.
    g. Software should not clear AUTO_CAL_ENABLE in SDMMC_AUTO_CAL_CONFIG_0 register after calibration. It should be set to 1 always to use calibration codes generated by calibration controller.
    h. If timeout occurs in auto-calibration process, software should program below drive strength code values. Software should clear AUTO_CAL_ENABLE in SDMMC_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.
    i. For 3.3V I/O:
        • APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_PAD_CAL_DRVUP = 0xC
        • APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_PAD_CAL_DRVDN = 0xC
    j. For 1.8V I/O:
        • APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_PAD_CAL_DRVUP = 0xB
        • APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_PAD_CAL_DRVDN = 0xF
    k. Enable periodic calibration to compensate for drift in driver strength due to temperature variation.
        • Re-run calibration for every 100 ms.
        • Start 100 ms timer after completing calibration iteration.
    - Once 100 ms timer expires, software has to re-initiate calibration as per above sequence but software needs to check if there are any ongoing transfers or not.
        • If there are no transfers, calibration can be triggered immediately or before a new data transfer starts. If a new data transfer request comes from upper OS layers during calibration, software should stall that request till calibration is completed.
        • If there are ongoing transfers, software should wait for transfers to complete and re-initiate calibration

25. Enable SD device clock by writing into SD Host registers (should follow SD bus power and Clock supply sequence described in section 3.2 of SD Host 4.0 specification). SD bus power field in Power Control register should be set to 1 before enabling SDCLK. See the SD clock supply sequence flowchart in Figure 152.
26. Do SD/SDIO device initialization as per the sequence mentioned in section 3.6 of SD Host 4.0 specification published by SDA. (SD Card Association).
27. Points to note:
    a. During device initialization, when I/O power is switched to 1.8V from 3.3V, software need to run auto-calibration again to set proper drive strengths before accessing external device.
    b. During voltage switching, follow the guidelines below to avoid pad damage.
        - Make sure APBDEV_PMC_PWR_DET_VAL_0_SDMMC3 = 1 when I/O voltage is 3.3V.
        - Program PMIC to switch I/O voltage to 1.8V from 3.3V.
        - Wait for 1 ms to cover LDO ramp time from 3.3V to 1.8V (5 mV/us)
        - Set APBDEV_PMC_PWR_DET_VAL_0_SDMMC3 = 0 since I/O voltage is switched to 1.8V.
        - Note I/O power switching can be done by programming PMU/PMIC along with programming SD Host Power Control register, 1.8V signaling enable in Host Control2 register.
        - Program vendor registers and pad controls for 1.8V I/O as mentioned in previous sections, if needed.
28. Software driver needs to set inbound clock trimmer tap value as per below guidelines to fix sampling point for data coming from device.
    a. In non-tunable modes (DS, HS, SDR12, SDR25 and DDR50 – up to 50 MHz), software needs to program inbound trimmer tap value as mentioned in Inbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL).
    b. To do data transfers in tunable modes (SDR50 and SDR104), Tuning procedure needs to be run to fix sampling point first. For Tuning procedure details, refer to Section 32.8.9: SDR50/SDR104/HS200 Tuning Procedure.
29. Now, Host and device are ready for doing a data transfers.
30. Follow software programming sequences mentioned in Chapter 3 of SD Host 4.0 specification published by SDA for doing data/cmd transfers.

32.7.2.4 SDMMC4 Initialization Sequence

This section provides SDMMC4 controller initialization sequence which should be followed by software driver before doing any CMD/DAT transfers to external eMMC device.

Power ON

1. Make sure SDMMC4 controller is powered up by programming external PMU/PMIC (platform specific) to set VDD_CORE voltage.
2. Refer to the PMU/PMIC datasheet and power up sequence section in TRM for programming details.

Program Clocks and Reset

Refer to Chapter 5: Clock and Reset Controller for full details of PLLs programming which are used for SDMMC4 clocking.

3. Set SDMMC4 reset:
    a. Keep SDMMC4 in reset by writing

    CLK_RST_CONTROLLER_RST_DEV_L_SET_0_ SET_SDMMC4_RST with ENABLE (0x1).
4. Program CLK divider and source
    a. Program SDMMC4 host clock divider
        - (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0_ SDMMC4_CLK_DIVISOR) and PLL source
        - (CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0_ SDMMC4_CLK_SRC) as per below guidelines.
    b. PLLP and PLLC4 are the main CLK sources for SDMMC4 controller.

- PLLP: VCO running at fixed 408 MHz
- PLLC4: VCO running at fixed ~1GHz (998.4 MHz +-18ppm quality) - Dedicated PLL for eMMC5.0 HS400 and HS667 clocking.
- For eMMC HS400/HS533 modes, software should use PLLC4 only since PLLC4 clock path is optimized to meet HS400 duty cycle distortion specification.

c. Software driver should program SDCLK Frequency select in SD host clock control register to get desired eMMC device clock which is derived from SDMMC4 host clock. Follow SD clock supply sequence mentioned in SD Host specification 4.0.

d. Maximum operating host clock frequency for SDMMC4 controller is 333 MHz.

e. The initializing frequency up to 400 kHz is provided to SD/SDIO card using the SDCLK Frequency select in SD host clock control register of SDMMC4 (current maximum divisor 2046 as per SD host specification). This is completely software dependent.

f. In HIGH SPEED DDR mode, SDMMC4 host should be run at 2X frequency than I/O. This requires SDMMC internal clock divider should be set to '2' [Standard Host Register SDMMCAB_SW_RESET_TIMEOUT_CTRL_CLOCK_CONTROL_0_ SDCLK_FREQUENCYSELECT value would be '1'].This is mandatory to make DDR50/52 mode work.

g. Note there is no SDCLK Frequency select restriction in SDR modes (DS, HS, SDR12, SDR25, SDR50, SDR104 and HS200).

h. HS400/HS533 mode: This is DDR mode only but has different timing and protocol compared to DDR50/52 mode. In this mode, host clock frequency should match device clock frequency. This is a Tegra design requirement. There is no specification compliance issue due to this as this mode is specific to eMMC devices and not defined in SD host specification.

The following table summarizes CLK divider options for PLLP and PLLC4 clock sources.

Table 206: PLL Clock Divider Options for PLLP and PLLC4

Speed Mode	Target I/O Max Frequency (MHz)	Clock Source[1]	Clock Source Frequency (MHz)	CAR divider[2]	SDMMC Host Clock Frequency (MHz)	SDMMC divider[3]	Actual I/O Frequency (MHz)
Identification	400 (kHz)	PLLP_OUT0	408	16.5	24.72727273	62	398.83 (kHz)
LEGACY SPEED	26	PLLP_OUT0	408	16	25.5	1	25.5
HIGH SPEED SDR	52	PLLP_OUT0	408	8	51	1	51
HIGH SPEED DDR	52	PLLP_OUT0	408	4	102	2	51
HS200 (SDR)	200	PLLC4_OUT2_LJ	199.68	1	199.68	1	199.68
HS400 (DDR)	200	PLLC4_OUT2_LJ	199.68	1	199.68	1	199.68
HS533 (DDR)	266	PLLC4_OUT0_LJ (DIVP=4)	249.6	1	249.6	1	249.6

1. It is set by SDMMC4_CLK_SRC in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0.
2. It is set by SDMMC4_CLK_DIVISOR in CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0.
3. It is set by SDCLK Frequency Select in Clock Control Register

5. Program SDMMC4 timeout clock source and divider

a. SDMMC4 uses 12 MHz TMCLK which is advertised in Host capabilities register -SDMMCAB_CAPABILITIES_0_ TIMEOUT_CLOCK_FREQUENCY.

b. Software should program below registers to provide TMCLK to SDMMC4.

- CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_ SDMMC_LEGACY_TM_CLK_SRC
- CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC_LEGACY_TM_0_ SDMMC_LEGACY_TM_CLK_DIVISOR

c. PLLP_OUT0 is the recommended TMCLK source for SDMMC4.

6. Enable SDMMC4 host clock and TMCLK by setting below registers.
   a. Enable SET_CLK_ENB_SDMMC4 in CLK_RST_CONTROLLER_CLK_ENB_L_SET_0.
   b. Enable CLK_ENB_SDMMC_LEGACY_TM in CLK_RST_CONTROLLER_CLK_OUT_ENB_Y_0.
7. Wait for at least 100 SDMMC4 host clock cycles time to get reset propagated down the pipe and also to get internal clocks stabilized.
8. Clear SDMMC4 reset by writing CLK_RST_CONTROLLER_RST_DEV_L_CLR_0_ CLR_SDMMC4_RST with ENABLE. Enable takes effect immediately.

Power ON eMMC or SDIO Device

9. Power ON eMMC (1.8V I/O). No need to wait for card detection as it is an embedded device.

PINMUX and eMMC IOBRICK Control Programming

10. SDMMC4 uses IOBRICK which is not pinmuxed. No PINMUX programming is needed.
11. E_INPUT and E_LPBK for SDMMC4 CLK pin should be enabled to provide loopback CLK for SDMMC4 Rx. If disabled, reads from SD/eMMC device won't work.
    a. APB_MISC_GP_EMMC4_PAD_CFG_CONTROL_0_EMMC4_PAD_E_DEEP_LPBK_CLK = 1'b1
    b. APB_MISC_GP_EMMC4_PAD_CFG_CONTROL_0_ EMMC4_PAD_E_INPUT_CLK = 1'b1
12. Program below register fields in APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0 register.
    a. APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC4_PAD_DRVUP_COMP = 0x10
    b. APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC4_PAD_DRVDN_COMP = 0x10
    c. Ensure that other bits in register APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0 are not touched by software. Default values should be used.
13. Select 50 Ohm driver:
    a. Ensure that the SDMMC4_COMP pad is connected to GND via 50 Ohm resistor on board.
    b. Make sure all DRV_TYPE fields in APB_MISC_GP_EMMC4_PAD_DRV_TYPE_CFGPADCTRL_0 are not updated by software. Software should use default values (DRV_TYPE_2X) only.
14. Pull Up/Pull Down values in APB_MISC_GP_EMMC4_PAD_PUPD_CFGPADCTRL_0 should match with default values. No updates are needed.

Program Host Vendor Registers

15. Program below registers before running auto-calibration to prevent CRC errors during data transfers done later.
    a. Write 0x1 into bit 19 (SPARE_OUT[3]) of register SDMMCAB_IO_SPARE_0.
    b. Write 0x0 into bit 2 (SEL_VREG) of register SDMMCAB_VENDOR_IO_TRIM_CNTRL_0. Refer to Section 32.8.5: DLL and Inbound Clock Trimmer Power Supply (VREG) Programming for software sequence to be followed to program this register field.
16. Set outbound clock trimmer tap value (works in all speed modes - up to 200 MHz) before doing any cmd/data transfers to device. This is needed to meet interface timing specification.
    a. SDMMCAB_VENDOR_CLOCK_CNTRL_0_ TRIM_VAL = 0x8
17. Set internal DQS trimmer tap value before doing any cmd/data transfers to device in HS400/533 mode. This is needed to meet interface timing specification.
    a. For device clock frequency <=200 MHz:

       SDMMCAB_VENDOR_CAP_OVERRIDES_0_ DQS_TRIM_VAL = 40 (decimal)
    b. For device clock frequency > 200MHz and <=333 MHz:

       SDMMCAB_VENDOR_CAP_OVERRIDES_0_ DQS_TRIM_VAL = 24 (decimal)

18. Set inbound clock trimmer tap value (works up to 50 MHz – in non-tunable modes such as SDR@26, SDR@52 and HIGH SPEED DDR) before doing any cmd/data transfers to device. Tuning procedure needs to be run in other modes – HS200 and HS400/533.
    a. SDMMCAB_VENDOR_CLOCK_CNTRL_0_ TAP_VAL = 0x0
19. Set Calibration pad VSEL:

    SDMMCAB_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7

Run Auto-Calibration

Run auto-calibration procedure to set proper drive strength codes for SDMMC4 IOBRICK. This is needed to use 50 Ohm driver for driving eMMC interface. This procedure should be run before doing device initialization.

20. Program drive code offsets for 1.8V operation before running auto-calibration:
    a. SDMMCAB_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8'd5
    b. SDMMCAB_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8'd5
21. Run auto-calibration procedure. Ensure that the SD/eMMC device clock is off during calibration procedure.
    a. Set E_INPUT_OR_E_PWRD = 1 in SDMMCAB_SDMEMCOMPPADCTRL_0 register, if not set.
    b. Wait for 1us after E_INPUT_OR_E_PWRD is enabled.
    c. Set AUTO_CAL_START and AUTO_CAL_ENABLE to 1 in SDMMCAB_AUTO_CAL_CONFIG_0 register.
    d. Wait for 1us.
    e. Wait for up to 10 ms for AUTO_CAL_ACTIVE in SDMMCAB_AUTO_CAL_STATUS_0 register to become 0. If it is not 0 after 10 ms, software can assume auto-calibration has timed out. If it becomes zero within 10 ms, it is assumed that calibration has completed.
    f. Clear E_INPUT_OR_E_PWRD to save power.
    g. Software should not clear AUTO_CAL_ENABLE in SDMMCAB_AUTO_CAL_CONFIG_0 register after calibration. It should be set to 1 always to use calibration codes generated by calibration controller.
    h. If timeout occurs in auto-calibration process, software should program the drive strength code values below. Software should clear AUTO_CAL_ENABLE in SDMMCAB_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.
    i. For 1.8V I/O:
        - APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC4_PAD_DRVUP_COMP = 0x10
        - APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0 _CFG2TMC_EMMC4_PAD_DRVDN_COMP = 0x10
22. Enable eMMC device clock by writing into SD Host registers (should follow SD bus power and Clock supply sequence described in section 3.2 of SD Host 4.0 specification). See the SD clock supply sequence flowchart in Figure 152. SD Bus Power register field in Power Control register should be set to 1 to enable SDCLK.
23. Do eMMC device initialization as per the sequence mentioned in eMMC5.0 specification published by JEDEC.
24. Run DLL calibration first, if software wants to use HS400/HS533 mode for doing data transfers to/from eMMC. For HS400 mode switching details, refer to Section 32.8.1: HS400 and HS667 Mode Selection.
25. Software driver needs to set inbound clock trimmer tap value as per below guidelines to fix sampling point for data coming from device.
    a. In non-tunable modes (LEGACY SPEED, HIGH SPEED SDR, and HIGH SPEED DDR – up to 52 MHz), software needs to program inbound trimmer tap value as mentioned in Inbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL).
    b. To do data transfers in tunable modes (HS200 and HS400), Tuning procedure needs to be run to fix sampling point first. For Tuning procedure details, refer to Section 32.8.9: SDR50/SDR104/HS200 Tuning Procedure.
26. Now, Host and device are ready for doing a data transfers.

27. Follow software programming sequences mentioned in Chapter 3 of SD Host 4.0 specification published by SDA for doing data/cmd transfers.

32.7.3 General Guidelines

These settings are applicable for all SD/MMC controllers in use:

- Enable SDR50 mode tuning support in SDMMC controller by setting SDMMCxx_VENDOR_CLOCK_CNTRL_0_SDR50_TUNING_OVERRIDE. This enables SDR50 mode tuning circuit in Host controller.
- Enable HS200 mode by setting UHS_MODE_SEL to SDR104 in HOST_CONTROL_2 (offset 03Eh) register. This field is used for SDR104 by standard Host register specification. Reusing the same for HS200 mode as there is no standard specification for eMMC Host.
- Enable HS400 (HS400 or HS667 or HS533 for eMMC5.0) support by setting UHS_MODE_SEL to HS400 in HOST_CONTROL_2 (offset 03Eh) register. This field is reserved as per standard Host register specification. But using the same as HS400 mode enable as there is no standard specification for eMMC Host.
- Set SDMMCxx_VENDOR_CLOCK_CNTRL_0_PADPIPE_CLKEN_OVERRIDE with OVERRIDE in all SD/eMMC speed modes.
- Write ‘1’ into bit 19 (SPARE_OUT[3]) of register SDMMCxx_IO_SPARE_0.
- Write ‘0’ into bit 2 (SEL_VREG) of register SDMMCxx_VENDOR_IO_TRIM_CNTRL_0.

32.7.3.1 SDCLK Divider (SDCLK Frequency select) Restriction

- In an SDR modes (DS, HS, SDR12/eMMC LEGACY SPEED, SDR25/eMMC HIGH SPEED SDR, SDR50, SDR104 and HS200), there is no restriction on SDCLK divider. Software can select any SDCLK divider to get desired device clock from Host clock. But Host clock should be less than the frequency that we support. Maximum Host clock frequency is 208 MHz for all controllers except in HS533 mode. In HS533 mode, they can use host clock of 267 MHz.
- In DDR50/52 mode, we have clock divider restriction. Host clock should be running at 2X speed than I/O clock. This is a design limitation. That's why we unadvertised DDR50 support (by default) in SD Host capabilities not to run into specification compliance issues. In this case software should use SDR50 instead DDR50 as both give same throughput. But it is not a specification compliance issue for eMMC as there is no eMMC host specification. In either case, software can always use DDR50/52 mode with the given SDCLK divider restriction.
- HS400/HS533 mode: This is DDR mode only but has different timing and protocol compared to DDR50/52 mode. In this mode, host clock frequency should match I/O clock frequency. This is our design requirement. There is no specification compliance issue due to this as this mode is specific to eMMC devices and not defined in SD host specification.

32.7.3.2 Card Insertion Interrupt

If software resets the SD/eMMC controller, it should not try to clear the Card Inserted interrupt within 300 us after asserting the SD/e.MMC reset. Trying to clear the Card Inserted interrupt before that time will not successfully clear the Card Inserted interrupt, causing an interrupt storm that may last for up to 300 us.

32.7.3.3 SDMMC1

These settings are specific to the SDMMC1 controller:

- Software should run auto-calibration procedure to setup proper driver for SD/eMMC interface before doing any transfers. Auto-calibration procedure determines and programs drive codes of pads, if it passes.
- If auto-calibration procedure fails due to some unknown reason, the following drive strength codes need to be programmed for the pads. Software should clear AUTO_CAL_ENABLE in SDMMCA_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.

Supply	33 Ohm	50 Ohm
	Up DN	Up DN

Supply	33 Ohm		50 Ohm	
3.3V	12	12	8	8
1.8V	11	15	3	4

DRV UP code - APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC1_PAD_CAL_DRVUP

DRV DN code - APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC1_PAD_CAL_DRVDN

- APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_ CFG2TMC_SDMMC1_CLK_CFG_CAL_DRVUP_SLWF = 0x1
- APB_MISC_GP_SDMMC1_PAD_CFGPADCTRL_0_ CFG2TMC_SDMMC1_CLK_CFG_CAL_DRVDN_SLWR = 0x1
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- For 3.3V operation:
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8’d125
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8’d0
- For 1.8V operation:
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8’d123
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8’d123
- E_INPUT and E_LPBK for SDMMC1 CLK pin should be enabled to provide loopback CLK for SDMMC1 Rx. If disabled, reads to SD/eMMC device won’t work.
  - APB_MISC_GP_SDMMC1_CLK_LPBK_CONTROL_0_SDMMC1_CLK_PAD_E_LPBK = 1’b1
  - PINMUX_AUX_SDMMC1_CLK_0_ E_INPUT_ ENABLE
- If using the SDMMC1 interface, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_SDMMC1_CLK_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOhm pull-up is used on the CMD line):
  - PINMUX_AUX_SDMMC1_CMD_0_PULL_UP_NORMAL
- If using SD or SDIO:
  - PINMUX_AUX_SDMMC1_CMD_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT3_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT2_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT1_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT0_0_PUPD_PULL_UP
- Select 33 ohm driver for SDMMC1 interface.
  - PINMUX_AUX_SDMMC1_CLK_0_ DRV_TYPE_DRIVE_2X
  - PINMUX_AUX_SDMMC1_CMD_0_ DRV_TYPE_DRIVE_2X.
  - PINMUX_AUX_SDMMC1_DAT0_0_ DRV_TYPE_DRIVE_2X.
  - PINMUX_AUX_SDMMC1_DAT1_0_ DRV_TYPE_DRIVE_2X.
  - PINMUX_AUX_SDMMC1_DAT2_0_ DRV_TYPE_DRIVE_2X.
  - PINMUX_AUX_SDMMC1_DAT3_0_ DRV_TYPE_DRIVE_2X.
- Make sure SDMMC1_COMP pad is connected to GND via 66 ohm resistor on board.
- DRV_TYPE_DRIVE2X and 66 Ohm calibration resistor are used to get drive strength codes for 33 ohm driver. This is Tegra specific implementation.

- Schmitt trigger selection for CLK/CMD/DAT pads
  - PINMUX_AUX_SDMMC1_xxxx_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock
  - PINMUX_AUX_SDMMC1_xxxx_0_ E_SCHMT_DISABLE for 3.3V I/O
  - [xxxx = CLK, CMD, DAT0, DAT1, DAT2 or DAT3].
- E_HSM selection for CLK/CMD/DAT pads
  - PINMUX_AUX_SDMMC1_xxxx_0_ E_HSM_DISABLE for both 3.3V and 1.8V I/O

    [xxxx = CLK, CMD, DAT0, DAT1, DAT2 or DAT3].

32.7.3.4 SDMMC2

These settings are specific to the SDMMC2 controller:

- Software should run auto-calibration procedure to setup proper driver for SD/eMMC interface before doing any transfers. Auto-calibration procedure determines and programs drive codes of pads, if it passes.
- If auto-calibration procedure fails due to some unknown reason, the following drive strength codes need to be programmed for the pads. Software should clear AUTO_CAL_ENABLE in SDMMCAA_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.

  UP - APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0_ CFG2TMC_EMMC2_PAD_DRVUP_COMP = 0x10

  DN - APB_MISC_GP_EMMC2_PAD_CFGPADCTRL_0_ CFG2TMC_EMMC2_PAD_DRVDN_COMP = 0x10
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8’h5
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8’h5
- Ensure that the SDMMC2_COMP pad is connected to GND via 50 ohm resistor on board.
- DRV_TYPE_DRIVE2X and 50 Ohm calibration resistor are used to get drive strength codes for 50 ohm driver. This is Tegra specific implementation.
- SDMMC2 controller uses EMMC IOBRICK which is not shared with any other controller. If using the SDMMC2 controller’s external interface with eMMC or SDIO, correctly enable the integrated pullup/down resistors.
- E_INPUT and E_LPBK for SDMMC2 CLK pin should be enabled to provide loopback CLK for SDMMC2 Rx. If disabled, reads to SD/eMMC device won’t work.

  APB_MISC_GP_EMMC2_PAD_CFG_CONTROL_0_EMMC2_PAD_E_DEEP_LPBK_CLK = 1’b1

  APB_MISC_GP_EMMC2_PAD_CFG_CONTROL_0_ EMMC2_PAD_E_INPUT_CLK = 1’b1

32.7.3.5 SDMMC3

These settings are specific to the SDMMC3 controller:

- Software should run auto-calibration procedure to setup proper driver for SD/eMMC interface before doing any transfers. Auto-calibration procedure determines and programs drive codes of pads, if it passes.
- If auto-calibration procedure fails due to some unknown reason, the following drive strength codes need to be programmed for the pads. Software should clear AUTO_CAL_ENABLE in SDMMC_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.

Supply	33 Ohm	50 Ohm
	Up DN	Up DN
3.3V	12 12	8 8
1.8V	11 15	3 4

DRV UP code - APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_PAD_CAL_DRVUP

DRV DN code - APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_PAD_CAL_DRVDN

- APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_CLK_CFG_CAL_DRVUP_SLWF = 0x1
- APB_MISC_GP_SDMMC3_PAD_CFGPADCTRL_0_CFG2TMC_SDMMC3_CLK_CFG_CAL_DRVDN_SLWR = 0x1
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- For 3.3V operation:
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8’d125
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8’d0
- For 1.8V operation:
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8’d123
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8’d123
- E_INPUT and E_LPBK for SDMMC3 CLK pin should be enabled to provide loopback CLK for SDMMC3 Rx. If disabled, reads to SD/eMMC device won’t work.
  - APB_MISC_GP_SDMMC3_CLK_LPBK_CONTROL_0_SDMMC3_CLK_PAD_E_LPBK = 1’b1
  - PINMUX_AUX_SDMMC3_CLK_0_ E_INPUT_ ENABLE
- If using the SDMMC3 interface, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_SDMMC3_CLK_0_PULL_UP_NORMAL
- If using eMMC (a stronger external 4.7 kOhm pull-up is used on the CMD line):
  - PINMUX_AUX_SDMMC3_CMD_0_PULL_UP_NORMAL
- If using SD or SDIO:
  - PINMUX_AUX_SDMMC3_CMD_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT3_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT2_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT1_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT0_0_PUPD_PULL_UP
- Select 33 ohm driver for SDMMC3 interface.
  - PINMUX_AUX_SDMMC3_CLK_0_ DRV_TYPE_ DRIVE_2X
  - PINMUX_AUX_SDMMC3_CMD_0_ DRV_TYPE_ DRIVE_2X.
  - PINMUX_AUX_SDMMC3_DAT0_0_ DRV_TYPE_ DRIVE_2X.
  - PINMUX_AUX_SDMMC3_DAT1_0_ DRV_TYPE_ DRIVE_2X.
  - PINMUX_AUX_SDMMC3_DAT2_0_ DRV_TYPE_ DRIVE_2X.
  - PINMUX_AUX_SDMMC3_DAT3_0_ DRV_TYPE_ DRIVE_2X.
- Ensure that the SDMMC3_COMP pad is connected to GND via 66 Ohm resistor on board.
- DRV_TYPE_DRIVE2X and 66 ohm calibration resistor are used to get drive strength codes for 33 ohm driver. This is Tegra specific implementation.
- Schmitt trigger selection for CLK/CMD/DAT pads
  - PINMUX_AUX_SDMMC3_xxxx_0_ E_SCHMT_ENABLE for 1.8V I/O - provides better duty cycle - low jitter clock
  - PINMUX_AUX_SDMMC3_xxxx_0_ E_SCHMT_DISABLE for 3.3V I/O

[xxxx = CLK, CMD, DAT0, DAT1, DAT2 or DAT3].

- E_HSM selection for CLK/CMD/DAT pads
  - PINMUX_AUX_SDMMC3_xxxx_0_ E_HSM_DISABLE for both 3.3V and 1.8V I/O

    [xxxx = CLK, CMD, DAT0, DAT1, DAT2 or DAT3].

32.7.3.6 SDMMC4

These settings are specific to the SDMMC4 controller:

- Software should run auto-calibration procedure to setup proper driver for SD/eMMC interface before doing any transfers. Auto-calibration procedure determines and programs drive codes of pads, if it passes.
- If auto-calibration procedure fails due to some unknown reason, the following drive strength codes need to be programmed for the pads. Software should clear AUTO_CAL_ENABLE in SDMMCAB_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in below registers and bypass calibration controller.
  - UP - APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0_CFG2TMC_EMMC4_PAD_DRVUP_COMP = 0x10
  - DN - APB_MISC_GP_EMMC4_PAD_CFGPADCTRL_0_CFG2TMC_EMMC4_PAD_DRVDN_COMP = 0x10
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 0x7
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8’h5
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8’h5
- Ensure that the SDMMC4_COMP pad is connected to GND via 50 ohm resistor on board.
- DRV_TYPE_DRIVE2X and 50 ohm calibration resistor are used to get drive strength codes for 50 ohm driver. This is Tegra specific implementation.
- SDMMC4 controller uses EMMC IOBRICK which is not shared with any other controller. If using the SDMMC4 controller’s external interface with eMMC or SDIO, correctly enable the integrated pullup/down resistors.
- E_INPUT and E_LPBK for SDMMC4 CLK pin should be enabled to provide loopback CLK for SDMMC4 Rx. If disabled, reads to SD/eMMC device won’t work.
- APB_MISC_GP_EMMC4_PAD_CFG_CONTROL_0_EMMC4_PAD_E_DEEP_LPBK_CLK = 1’b1
- APB_MISC_GP_EMMC4_PAD_CFG_CONTROL_0_ EMMC4_PAD_E_INPUT_CLK = 1’b1

32.7.4 SDR50 Mode Workaround for Avoiding Read CRC Errors

The SDMMC controller DMA buffer may become full during read data transfer and trigger an SD card clock stopping during block transfer (clock stopping is the flow control mechanism defined on SD interface). In SDR50 mode, this intra-block SD card clock stopping may cause the SDMMC controller to misalign the received data and result in a read CRC error.

The failure occurs only if:

1. Intra-block SD clock stopping occurs AND
2. The trimmer tap value selected for SDR50 read data receiver clock is more than one UI (one SD clock cycle). This issue is present only in SDR50 mode. Only SD/SDIO devices are affected with this issue. eMMC is not affected.

32.7.4.1 Software Workaround

Set UHS_MODEL_SEL to SDR104 in the SDMMC controller even though the SD/SDIO card is in SDR50 mode. This enables inter-block clock stopping in the SDR50 mode which prevents CRC errors.

32.8 Programming Guidelines for eMMC HS400/HS667 Mode

This section provides HS400/HS533/HS667 mode programming guidelines. This mode is eMMC specific mode and supported by only SDMMC2 and SDMMC4 controllers.

32.8.1 HS400 and HS667 Mode Selection

The software driver community is using SD host specification for developing eMMC driver. Tegra X1 SDMMC supports HS400 and HS667 speed modes for devices which support eMMC5.0 specification and beyond. But the SD host specification does not have support for these modes. Our SD/eMMC host controller uses RSVD fields of UHS_MODE_SEL to enable HS400 and HS667 modes which adheres to SD specification compliance.

Software driver should program UHS_MODE_SEL as per below to select appropriate speed modes.

32.8.1.1 Host Control2 Register / Auto CMD Error Status Register

UHS_MODE_SEL - UHS Mode Select

- 000b SDR12/eMMC LEGACY SPEED
- 001b SDR25/eMMC HIGH SPEED SDR
- 010b SDR50
- 011b SDR104 (SD/SDIO)/HS200(eMMC)
- 100b DDR50 (SD/SDIO)/HIGH SPEED DDR (eMMC)
- 101b HS400(eMMC)/HS667(eMMC)
- 111b UHS2
- 111b Reserved

32.8.2 Clock Divider Programming in HS400/667 Modes

SDCLK Frequency Select should be set to ‘0’ in HS400/HS667 modes. No other divisors are supported.

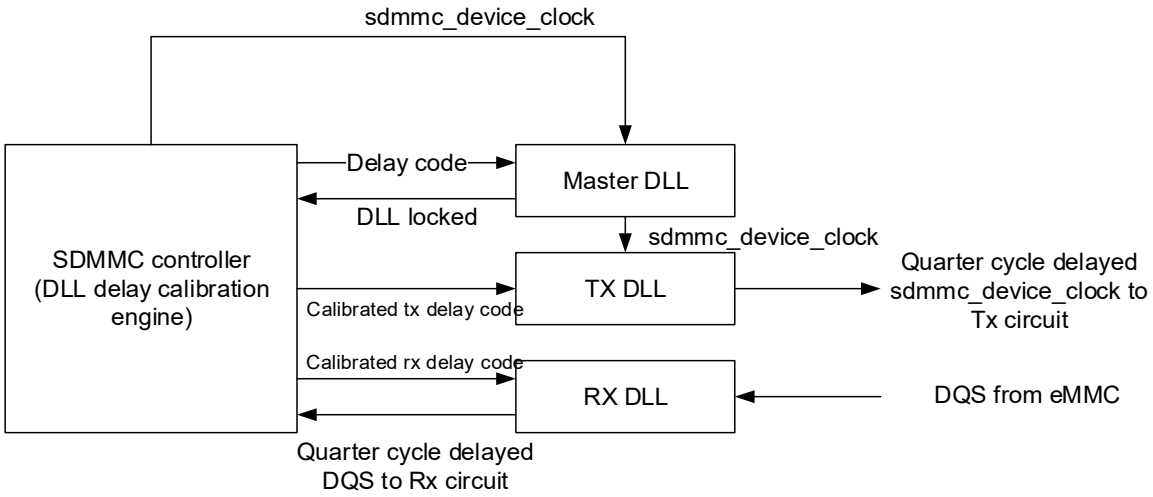
For example, to get 333 I/O clock frequency, sdmmc_clk should be running at 333 MHz only. That means both I/O and controller should be running at same clock frequency in HS400/HS667 modes.

32.8.3 DLL Calibration

HS400/533 transmitter and receiver circuit uses a pair of DLLs (Tx DLL and Rx DLL) to delay tx/rx clock by quarter cycle. Quarter cycle delay is needed as per our HS400 implementation. DLL is a mixed signal circuit whose delay code should be programmed to get quarter cycle delay. DLL delay code varies based on Si process and frequency of clock used. So, we use a Master DLL for calibration to get delay code required for getting quarter cycle delay at given frequency of operation. SD/MMC controller has DLL calibration controller which sends delay code in iterative fashion to Master DLL and gets feedback from it. As per the feedback received from Master DLL per applied delay code, DLL calibration controller determines the delay code needed to get quarter cycle delay. Hardware applies the determined code to both Tx and Rx DLLs and turns off Master DLL. This ends DLL calibration process.

Below diagram shows DLLs present in SD/MMC subsystem.

Figure 153: DLLs in SD/MMC Subsystem



Software needs to calibrate DLL before using eMMC5.0 HS400/HS667 mode (before tuning process). Re-calibration is required whenever eMMC I/O clock frequency is changed. This can be done as a part of standard clock frequency change sequence.

Points to note:

- DLL calibration should work at when target I/O CLK is running at ~200 (180 to 200) MHz and above.
- For Target I/O CLK frequency is below 180 MHz, software should calculate delay code as per below equation and program delay code:
  - Delay Code = (Delay Code at 200 MHz) * 200 MHz / Target I/O Frequency
  - Delay code at ~200 MHz is determined by running DLL calibration and the same is available for software in below RO registers.
    - Tx DLL: SDMMCxx_VENDOR_DLLCAL_CFG_STA_0_ TX_DLY_CODE_ADJ
    - Rx DLL: SDMMCxx_VENDOR_DLLCAL_CFG_STA_0_ RX_DLY_CODE_ADJ
- Under 85 MHz, DLL delay code may reach its maximum code (128 taps) for FF corner chips. In this case, software should use 128 taps.
- Software can run DLL calibration at 200 MHz once at boot time, save those values, scale and use them whenever frequency is changed. This is needed if software wants to do DFS (Dynamic Frequency Scaling).
- Program calculated delay code by writing into below registers:
  - Set SDMMCAB_VENDOR_DLL_CTRL0_0_ DLLCAL_BYPASS with 0x1 to bypass DLL calibration.
  - Set SDMMCAB_VENDOR_DLL_CTRL0_0_ TX_SLV_DLL_DLY_CODE with delay code calculated in above step.
  - SDMMCAB_VENDOR_DLL_CTRL0_0_ RX_SLV_DLL_DLY_CODE with delay code calculated in above step.
- PROD setting:
  - *ONLY* for HS533 mode at 266 MHz the following setting should be used to improve setup/hold time margins:

    TX_DLL_OFFSET = -4 (decimal negative four) = 0x7C

    (HS400 mode at 200 MHz or lower doesn't need the above setting)
  - Register field SDMMCxx_VENDOR_DLL_CTRL0_0_TX_DLY_CODE_OFFSET should be programmed with above value in two’s complement format.

32.8.4 Software Programming Sequence for Executing DLL Calibration

- Set SDMMCxx_VENDOR_IO_TRIM_CNTRL_0_SEL_VREG to 0x0 to select Band Gap VREG to supply DLL. If it is 0x1, set it to 0x0 and follow guidelines mentioned in Section 32.8.3: DLL Calibration before triggering DLL calibration process.

- Write DLL CAL CFG registers -
  - SDMMCAB_VENDOR_DLLCAL_CFG_0,
  - SDMMCAB_VENDOR_DLL_CTRL0_0,
  - SDMMCAB_VENDOR_DLL_CTRL1_0 and SDMMCAB_VENDOR_DLLCAL_CFG_STA_0 to update various control parameters, if software wants to override default values. It is strongly recommended to use default values for DLL calibration.
- Software can choose either using software timer (5 ms) or END_COUNT for stopping calibration.
- Program END_COUNT field in register SDMMCAB_VENDOR_DLLCAL_CFG_STA0_0 or maintain software timer for DLL lock time (5 ms). Default value of END_COUNT is sufficient.
- Set CALIBRATE bit in register SDMMCAB_VENDOR_DLLCAL_CFG_STA0_0 to trigger calibration process.
- If END_COUNT is set to zero, software should stop calibration by clearing CALIBRATE bit once software timer expires. Else poll for CALIBRATE = 0 (hardware clears it once it sees end of calibration condition.)
- Poll for CAL_ACTIVE=0 to see if calibrated delay code is applied to slave DLLs. DLL calibration process may take up to 10 ms.
- Software can exit DLL calibration loop after this and can start tuning process.

32.8.5 DLL and Inbound Clock Trimmer Power Supply (VREG) Programming

- Inbound clock trimmer is used to delay loopback clock to sample incoming data and command response in SDR50, SDR104, and HS200. Inbound clock trimmer is also used in HS400 without enhanced strobe mode for command response).
- DLL is used in HS400 receiver to latch incoming data and command response when enhanced strobe mode is enabled.
- Delay chain in trimmer and DLL have two power supplies.
  - Regulated power supply (VREG) from Band gap (BG) reference circuit which is a constant power supply has almost zero variation with VDD_CORE changes. This is the recommended power supply for both trimmer and DLL.
  - VDD_CORE/VAUXC is another option. This one is used when BG circuit is turned off in idle state to save power. Software should not use this during transfers as it causes drift in trimmer and DLL delay with VDD_CORE/VAUXC change.
- Software should use BG power supply to power both trimmer and DLL to make delay produced by internal delay chain invariant to Vcore changes.
  - By default, VDD_CORE/VAUXC (core power supply) is used as trimmer/DLL power supply which saves power by turning off Band gap regulator supply (BG). Software should set SDMMCxx_VENDOR_IO_TRIM_CNTRL_0_SEL_VREG to 0x0 in all speed modes to select VREG as trimmer power supply which makes trimmer delay independent of VDD_CORE.
- Software should follow below sequence for switching power supply from VDD_CORE/VAUXC to BG.
  - Make sure SDMMC is idle – no activity on SD/eMMC interface.
  - Turn off SD interface clock.
  - Program SDMMCxx_VENDOR_IO_TRIM_CNTRL_0_SEL_VREG with 0x0 to turn ON BG supply.
  - BG circuit power up time is ~3 us. Wait for 3 us to make sure BG supply is stabilized.
  - Both DLL and inbound clock trimmer are powered using BG supply and their output could glitch with this power supply switching to/from BG from/to VAUXC/VDD_CORE. This glitch can happen irrespective of input clock/DQS state which would cause disturbance in downstream Rx logic which in turn affects data transfers later on.
  - To prevent errors on interface due to DLL/trimmer output glitch, software driver should reset host controller by issuing both SW_RESET_DAT and SW_RESET_CMD (sequence should be followed as per SD host specification) before starting any new transfers.

- Software can choose to switch back to VAUXC from BG to turn OFF Band gap regulator to save power when eMMC/SD interface is idle. But software should switch back to BG before starting new transfers and follow above sequence for VAUXC to BG switching.

NOTE (informative):

DLL output glitch (not trimmer glitch) can be avoided by following below steps:

- Program below register fields first before switching power supply from/to BG to/from VAUXC.
  - SDMMCAB_VENDOR_DLLCAL_CFG_STA_0SLV_DLL_CLK_OUT_DIS_OVERRIDE_EN = 0x1
  - SDMMCAB_VENDOR_DLLCAL_CFG_STA_0SLV_DLL_CLK_OUT_DIS = 0x1
- Wait for 32 SDMMC clock cycles after power supply switching (above sequence).
- Clear override SLV_DLL_CLK_OUT_DIS_OVERRIDE_EN to give control to DLL calibration engine.
- This sequence can be used to clamp DLL output to zero whenever there is change in DLL controls.

- This sequence can help avoiding DLL glitch but not trimmer glitch. Software has to issue SW_RESET to recover from bad state due to trimmer glitch. To make fix simple and reduce software overhead, we suggest software to issue SW_RESET only to recover from DLL and trimmer glitch issues whenever there is a change in DLL and trimmer power supply.

32.8.6 HS400 Mode Guidelines

32.8.6.1 Software Programming Sequence for Entering into HS400 Mode

Refer to the eMMC5.1 specification for HS400 mode selection.

32.8.6.2 Software Programming Sequence for Changing the Frequency in HS400 Mode

- Follow the clock stop sequence defined in SD host software sequence chapter.
- Change the SDMMC Clk Frequency.
- SDCLK turn ON sequence.
- Perform DLL calibration.
- Configure device in HS400 mode.
- Set HS400 mode in the controller.

32.8.7 Clock Trimmer Settings

32.8.7.1 IB and OB Trimmer Tap Value Setting

Software should select proper tap values for:

- Outbound clock trimmer - SDMMC_VENDOR_CLOCK_CNTRL_0_TRIM_VAL and
- Inbound clock trimmer - SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL to prevent data/cmd CRC errors.

Note: *Trimmer settings depend on speed mode used. Program as per the tables below.*

Inbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TAP_VAL)

The tap values below can be used up to SDR25/DDR50 and HIGH SPEED DDR (eMMC) speeds (Frequency <= 52 MHz).

(Trace length = 0.45 ns for SDMMC2 and SDMMC4 and 0.75 ns for SDMMC1 and SDMMC3.)

Controller	Tap Value for I/O trimmer (default)	Tap Value for Core Trimmer (If SDMMC_VENDOR_SYS_SW_CNTRL_0_IO_TRIM_BYPASS is set to 1)
SDMMC1	4	3
SDMMC2	0	3
SDMMC3	3	3
SDMMC4	0	3

Notes:

- *These tap values work only up to 48 MHz in DDR50 mode for SDMMC1 and SDMMC3. Software should use SDR50, if it needs to run I/O at 50 MHz for SDMMC1 and SDMMC3.*
- *Tunable modes: Software should use tuning procedure to determine the tap value in SDR50 and SDR104/HS200 modes.*

Outbound Tap Values (SDMMC_VENDOR_CLOCK_CNTRL_0_TRIM_VAL)

Below settings are valid in all speed modes.

Controller	Tap Value
SDMMC1	2
SDMMC2	8
SDMMC3	3
SDMMC4	8

DQS Trimmer Tap Values (SDMMC_VENDOR_CAP_OVERRIDES_0_ DQS_TRIM_VAL)

HS400 (<=200 MHz):.

Controller	Tap Value
SDMMC1	N/A
SDMMC2	40
SDMMC3	N/A
SDMMC4	40

HS533/667 (>200 MHz and <=333 MHz):

Controller	Tap Value
SDMMC1	N/A
SDMMC2	24
SDMMC3	N/A
SDMMC4	24

32.8.8 Pad Control Programming for SDMMC1/SDMMC3 to Avoid Pad Damage

- In Tegra X1, PWT_DET cells are removed for non-bootable interfaces. software should set PWR_DET value appropriately by programming PMC register whenever Vio is changed from 3.3V to 1.8V and vice versa.
- SDMMC1 and SDMMC3 use BDSDMEM pads which are capable of supporting both 3.3V and 1.8V I/O. 3.3V would be selected on PWR_ON.

- Before ramping up to 3.3V on VDDIO_SDMMC1/SDMMC3 rails, pad voltage must be set correctly. Set PWR_DET_VAL_0 = 1 and SDMMCx_SDMEMCOMPPADCTRL_0_PAD_E_INPUT_OR_E_PWRD = 1. Not doing so can damage the I/O PADS and result in higher leakage.
  - PWR_DET_VAL=1 => Enable E_33V for 3.3V I/O voltage
  - PWR_DET_VAL=0 => Disable E_33V for 1.8V I/O voltage
  - Pad damage may occur if PWR_DET_VAL = 0 when VDDIO_SDMMC = 3.3V.
  - PMC register:
    - APBDEV_PMC_PWR_DET_VAL_0_SDMMC3 for SDMMC3
    - APBDEV_PMC_PWR_DET_VAL_0_SDMMC1 for SDMMC1

32.8.8.1 Software Sequence for SD Card

On Cold boot / Card Insertion (via GPIO interrupt) / LP0 resume:

- VDDIO_SDMMC1 or VDDIO_SDMMC3 is tuned OFF.
- Set PAD_E_INPUT_OR_E_PWRD = 1 of SDMMCx_SDMEMCOMPPADCTRL_0[31] to avoid leakage power and pad damage.
- Set PWR_DET_VAL = 1 (Enable E_33 V)
- Turn on 3.3V slot power at SD Card Slot
- Turn on VDDIO_SDMMC 3.3V I/O at Tegra
- Leave PAD_E_INPUT_OR_E_PWRD =1 until auto_calibration is completed for 3.3 V level.
- Clear PAD_E_INPUT_OR_E_PWRD =0 when auto_calibration is completed
- Initialization sequence
- If card support UHS-1 modes issue voltage switch command
- Change I/O Voltage from 3.3V to 1.8V
- Set PWR_DET_VAL = 0 (Disable E_33 V)

On Card removal (via GPIO interrupt):

- Enable GPIO mode and output "1" on the entire SDMMC bus (CLK, CMD, DAT3, DAT2, DAT1, and DAT0).
- Shutdown the VDDIO_SDMMC rail
- Shutdown the SDMMC slot power (3.3V supply)
- Set SDMMCx in APBDEV_PMC_NO_IOPOWER_0 as SDMMCx interface is shutdown
- Set E_33V = 0 via PWR_DET_VAL = 0
- Set E_33V = 1 (default POR) via PWR_DET_VAL = 1
- Disable GPIO mode and set pinmux to POR values

32.8.8.2 Software Sequence for SDIO

For 1.8V SDIO: Set PWR_DET_VAL = 0 (Disable E_33V) always.

32.8.9 SDR50/SDR104/HS200 Tuning Procedure

32.8.9.1 Vendor Registers Programming Before Executing Tuning Procedure

- SDMMC_VENDOR_IO_TRIM_CNTRL_0_SEL_VREG should be set to 0x0 to select VREG as trimmer power supply which makes trimmer delay independent of VDD_CORE. By default, VDD_CORE is used as trimmer power supply

which saves power by turning off BG+REG. For switching power supply, software should guidelines mentioned in Section 32.8.5: DLL and Inbound Clock Trimmer Power Supply (VREG) Programming.

- SDMMC_VENDOR_SYS_SW_CNTRL_0_ IO_TRIM_BYPASS should be set to 0x0 (default value).
- Set SDMMC_VENDOR_TUNING_CNTRL0_0_ NUM_TUNING_ITERATIONS based on speed mode selected.
  - TRIES_256 for SDR50 mode
  - TRIES_128 for SDR104 mode
  - TRIES_128 for HS200 and HS400 modes
  - TRIES_64 for HS533 and HS667 modes
- Set SDMMC_VENDOR_TUNING_CNTRL1_0_ STEP_SIZE_SDR104_HS200 with 0x0
- Set SDMMC_VENDOR_TUNING_CNTRL1_0_ STEP_SIZE_SDR50 with 0x0

32.8.9.2 Tuning Procedure

1. Make sure SDCLK is running at desired frequency as per speed mode selected for future data transfers.
2. Set HostControl2.ExecuteTuning = 1
3. Set ClockControl.SDClockEnable = 0 to stop SDCLK
4. Issue Tuning command (CMD19 for SD/SDIO or CMD21 for eMMC) as per CMD issuing sequence defined in SD host specification.
5. Wait for 1 us
6. Issue SW_RESET_FOR_DAT_LINE && SW_RESET_FOR_CMD_LINE by programming SDMMC_SW_RESET_TIMEOUT_CTRL_CLOCK_CONTROL_0 register.
7. Wait for SW_RESET for both CMD and DAT lines to complete.
8. Set ClockControl.SDClockEnable = 1 to enable SDCLK
9. Wait until BufferReadReady is 1
10. Check ExecuteTuning:

    if 0: Go to STEP 11
    if 1: Go back to STEP 3
11. Check SamplingClockSelect:

    if 0: Tuning Failure
    if 1: Tuning Success
12. Now Host is ready for data transfers.

Note: *In tunable modes, hardware tuning engine sets trimmer tap value but in non-tunable modes (during speed mode change or LP0 exit or DFS), software driver has to update trimmer tap value manually. In that case also, SW_RESET should be issued to host to reset internal logic.*

32.8.9.3 Post Tuning Procedure

When connected to devices that drive fast edges, the SOC SDMMC controller may not be able to detect a failing tap between two valid windows during tuning. This causes the SDMMC controller to set a marginal inbound tap delay which is at the middle of two merged valid windows. This can result in data or command CRC during SDMMC transactions. This affects all the tunable SDMMC speed modes (SDR50, SDR104, HS200, and HS400 without enhanced strobe mode). As an issue workaround, it is recommended that software implement a post-tuning function after tuning is completed and to apply the correct tap value when a merge window is detected. Refer to the “SDMMC Tuning Update” application note for details.

32.8.9.4 Tuning Bug Workaround

As a part of a workaround, the trimmer input clock is stopped during tap value change which would avoid output to glitch in most cases. In some Vreg and Vcore combinations and at high frequencies, the trimmer may glitch even though the input clock is not running when the tap value is changed.

To nullify the effect of a glitch on downstream logic which uses the trimmer output clock, issue a SW_RESET to clear the data pipeline after the tap value is changed.

Workaround Sequence for Tuning

STEP_SIZE used for the tuning process should be set to 0x1 (register value is 0x0) for this workaround.

- Stop SD/eMMC device clk which would gate input clk to trimmer.
- Issue Tuning Command to Host controller (this would trigger trimmer tap value update)
- Wait for 1 us - trimmer output uncertainty period – trimmer output could glitch and corrupt downstream logic.
- Recover from bad state caused by trimmer glitch by issuing SW_RESET for both CMD and DAT circuits.
  - Issue software reset for DAT Line and software reset for CMD line in Software Reset register to 1.
  - Wait for SW_RESET for both CMD and DAT lines to complete.
- Start SD/eMMC (SDCLK) device clk
- Wait for buffer_read_ready=1 (Tuning command completion interrupt)
- Go back to step 1 if tuning process is not done.

Workaround for Other Scenarios

SW_RESET should be issued whenever software changes trimmer tap value during speed mode change or LP0 exit or DFS.

Number of Tuning Iterations

As per the software workaround, software should use 256 tuning iterations for scanning all 256 taps of trimmer. But it is not needed for every I/O speed mode. Software can use fewer iterations based on speed mode selected (I/O clock frequency) to reduce tuning time. Below register field should be written with appropriate value per speed mode.

SDMMC_VENDOR_TUNING_CNTRL0_0 (Vendor Tuning Control0 register), Offset: 0x1c0

15:13	TRIES_40	NUM_TUNING_ITERATIONS:The number of tuning iterations to be used by tuning circuit; Preferred value is 40 iterations as per Host specification. 0 = TRIES_40 1 = TRIES_64 (for HS533 and HS667 modes) 2 = TRIES_128 (for SDR104, HS200, and HS400 modes) 3 = TRIES_192 4 = TRIES_256 (for SDR50 mode)

32.8.10 Pad Drive Strength Calibration Sequence

SDMMC controller pads need calibration to set proper drive strength for 33 ohm (SDMMC1/3) / 50 ohm (SDMMC2/4) driver. Calibration process uses a dedicated calibration pad which is connected to a 66 ohm (SDMMC1/3) / 50 ohm (SDMMC2/4) resistor to ground. Pad control DRV_TYPE is should select DRIVE2X to get 33 ohm/ 50 ohm driver codes using 66 ohm / 50 ohm calibration resistor. This is a Tegra specific implementation. Programming details are given in initialization sequence.

32.8.10.1 Drive Strength Calibration Process

In order to maintain a constant drive strength of 33/50 ohms for SD or eMMC interface, Tegra X1 uses an additional calibration pad (SDMMCxx_COMP_PD), which have the same driver as the other pads, to account for

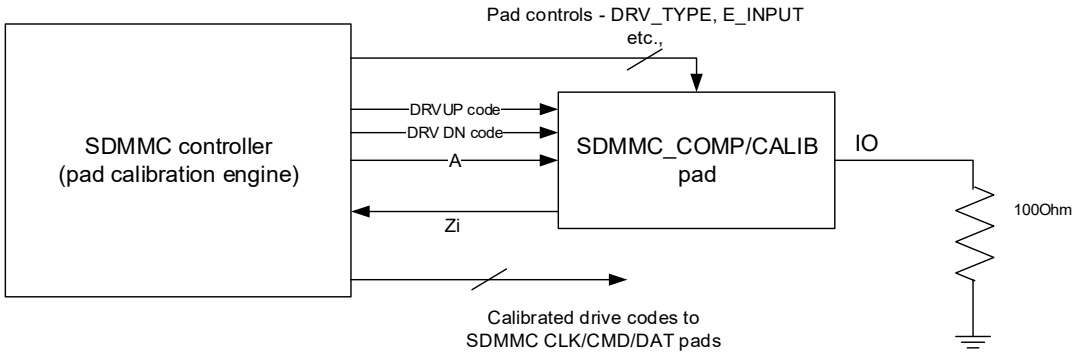
1. Process, Voltage and Temperature (PVT) variations - resistor calibration is done once during boot.
2. Temperature Drifts, resistor calibration needs to be done when pads are idle and dynamically

For this, SDMMC implements resistor calibration registers. The calibration can either be done manually by the Host Driver or SDMMC can do Auto Calibration. The basic idea is to keep increasing the voltage and identifying the correct level for driving a 1 and a 0 since PVT/Temperature will vary the drive impedance. This is done with the help of an external precise calibration resistor which is connected to the COMP_PD pad and ground.

Within the calibration pads, the drive codes are incremented in steps which, in turn, increase the voltage level on the COMP_PU and decreases the voltage levels on COMP_PD. An internal comparator compares the voltage with a reference voltage and drives a 1 on a match. At this point, the values on the DRVUP/DN values of the comp pads are sampled and programmed into the other pads.

Below diagram shows calibration circuit.

Figure 154: Calibration Circuit



32.8.10.2 Calibration Software Sequence

This section provides drive strength calibration sequence details. This sequence is required during initialization or switching frequencies or periodic intervals depending on the platform usage scenario. Ensure that the SD/eMMC device clock (SDCLK) is off during calibration procedure.

- Set up proper reference voltage as per the SDMMC port as described in Section 32.8.7: Clock Trimmer Settings.
- Set E_INPUT_OR_E_PWRD = 1 in SDMMCxx_SDMEMCOMPPADCTRL_0 register.
- Wait for 1us after E_INPUT_OR_E_PWRD is enabled.
- Set AUTO_CAL_START and AUTO_CAL_ENABLE to 1 in SDMMCxx_AUTO_CAL_CONFIG_0 register.
- Wait for 1 us.
- Wait for up to 10 ms for AUTO_CAL_ACTIVE in SDMMCxx_AUTO_CAL_STATUS_0 register to become 0. If it is not 0 after 10 ms, software can assume auto-calibration has timed out. If it becomes zero within 10 ms, it is assumed that calibration has completed.
- Clear E_INPUT_OR_E_PWRD to save power.
- Software should not clear AUTO_CAL_ENABLE in SDMMCxx_AUTO_CAL_CONFIG_0 register after calibration. It should be set to 1 always to use calibration codes generated by calibration controller.
- If software timeout occurs during calibration process, software should program drive strength up/down static values as described in the Run Auto-Calibration section. Software should clear AUTO_CAL_ENABLE in SDMMCxx_AUTO_CAL_CONFIG_0 register to use calibration codes programmed in APB_MISC_GP* registers and bypass calibration controller.

32.8.10.3 Pseudo Code for Calibration

1. auto_calibration() {
2. e_input = (*SD_SDMEMCOMPPADCTRL >> SDMMC_SDMEMCOMPPADCTRL_0_PAD_E_INPUT_OR_E_PWRD_FIELD)&1 ; // read e_input value

3. if(e_input == 0) { // enable e_input if not enabled yet
4. *SD_SDMEMCOMPPADCTRL |= 1 << SDMMC_SDMEMCOMPPADCTRL_0_PAD_E_INPUT_OR_E_PWRD_FIELD; // enable e_input
5. uSecDelay(1) ; } // Wait 1 us after e_input is enabled
6. *SD_AUTO_CAL_CONFIG |= (1<<31) | (1<<29); // enable auto_calibratoin and start it.
7. uSecDelay(1) ; } // Wait 1 us before polling CAL_ACTIVE status
8. while(*SD_AUTO_CAL_STATUS & (1<<31)) {} // wait for auto_calibration completed status
9. if(DISABLE_E_INPUT_AFTER_CALIB)
10. *SD_SDMEMCOMPPADCTRL &= ~(1 << SDMMC_SDMEMCOMPPADCTRL_0_PAD_E_INPUT_OR_E_PWRD_FIELD); }
11. Re-run calibration as per the conditions mentioned in the previous section.

32.8.11 eMMC Boot Modes

The section of this document on the boot process describes the boot process in more detail. Note that only the SDMMC4 controller can be used as the eMMC boot device.

32.8.12 Boot Option1

1. The clock to the card needs to be configured to 26 MHz.
2. Program the number of blocks. Configuring block length has no effect; it is always fixed to 512 bytes.
3. Configuring the data transfer direction has no impact, since it taken as CARD to HOST for boot mode.
4. Select either SDMA or PIO mode.
5. Configure SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0 – The maximum timeout value that needs to be programmed is 50ms.The value programmed should exclude the 74 cycles that are required to enter boot mode.
6. Configure SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0 – The maximum timeout value that needs to be programmed is 1 second. The value programmed should exclude the 74 cycles that are required to enter boot mode.
7. Configure SDMMC_VENDOR_BOOT_CNTRL_0[1] to 1 to ask the engine to look for boot_ack.
8. Configure SDMMC_VENDOR_BOOT_CNTRL_0[0] to trigger the engine.
9. If SDMMC_INTERRUPT_STATUS_0[31] = 1,the BOOT_ACK is not equal to 00101.The engine just informs the software and continues to fetch data from the card.
10. If SDMMC_INTERRUPT_STATUS_0[30] = 1, the BOOT_ACK timeout has occurred. The engine just informs the software and continues to fetch data from the card.
11. The software should look for transfer complete interrupt. If “data_time_out” error has occurred, then the data transfer is not successful.
12. After successful data transfer from card, the software should look for “sdma_engine_busy” bit of SDMMC_OBS_SDMMC_DBG0_0[0]. Software shouldn’t program commands until this bit is zero.

32.8.13 Boot Option2

Boot option2 is treated like any normal read command. The BOOT_ACK should be enabled if the card supports boot acknowledgment. SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0 and SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0 should be programmed based on the frequency of operation.

32.8.14 Card Detect and Write Protect

Only SDMMC1 and SDMMC3 support removable SD cards. The standard SD Host, SDCD# pin act as card detect pin and CDWP# pin act as Write protect pin. The SD Card State registers can be ignored for embedded on-board devices.

PIN MUX detail for CD and WP pins for SDMMC1 and SDMMC3.

PINMUX_AUX_SDMMC1_WP_N_0_PUPD_PULL_UP

PINMUX_AUX_KB_COL4_0_PUPD_PULL_UP - This is the Pinmux option for SDMMC1 CD

PINMUX_AUX_KB_COL4_0_PUPD_PULL_UP - This is the Pinmux option for SDMMC3 WP

PINMUX_AUX_SDMMC3_CD_N_0_PUPD_PULL_UP

Additionally, an abort command should be issued if a card is removed during an SD transfer.

32.8.15 SD3.0 Signal Voltage Switching

Note: *This section is applicable only to the removable-card-capable SDMMC1 and SDMMC3 controllers. The SDMMC2 and SDMMC4 controllers do not require these settings.*

32.8.16 SD Bus Power

When the standard SD driver changes the Power Control Register's SD Bus Power field, the hardware needs assistance from platform software to implement the change via the PMIC.

32.8.16.1 Initialization

Before the standard SD drive begins running:

1. Route the SD/MMC controller's system interrupt to the appropriate driver.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS to clear any latent interrupts.
3. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_ENABLE to enable the system interrupt on an SD3.0 UHS-I Bus Power change event.

32.8.16.2 Entry

Once an SD/MMC system interrupt is received:

1. Confirm the SDMMC System interrupt was from a Bus Power change event by reading SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS to clear the interrupt condition.
3. Read SD BUS POWER for VDD1 in POWER CONTROL register to determine the power state change request.

If Bus Power is cleared, power off the SD power supply via the PMIC. Otherwise, if Bus Power is set, power on the SD power supply via the PMIC.

32.8.16.3 Exit

No actions required.

32.9 SD/MMC Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

32.9.1 Standard Registers

Note: *For standard registers' usage, refer to the SD Host Controller Specification version 4.00.*

There are a few vendor-specific register fields within the standard register offset range (0x00 through 0xff). These fields are described below:

32.9.1.1 Interrupt Status Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0	VEND_SPEC_ERR Bit 1: BOOT_ACK_ERR: Occurs when Boot Ack Status is not equal to '010' Bit 0: BOOT_ACK_TIMEOUT_ERR: Occurs when Boot Ack is not received within the programmed number of cycles.

32.9.1.2 Interrupt Status Enable Register

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0xxx000000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR 0 = Enable neither BOOT_ACK_ERR nor BOOT_ACK_TIMEOUT_ERR statuses. 1 = Enable only BOOT_ACK_TIMEOUT_ERR status. 2 = Enable only BOOT_ACK_ERR status. 3 = Enable both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR statuses.

32.9.1.3 Interrupt Signal Enable Register

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0xxx000000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR 0 = Enable neither BOOT_ACK_ERR nor BOOT_ACK_TIMEOUT_ERR interrupts to CPU. 1 = Enable only BOOT_ACK_TIMEOUT_ERR interrupt to CPU. 2 = Enable only BOOT_ACK_ERR interrupt to CPU. 3 = Enable both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR interrupts to CPU.

32.9.1.4 SDMMC_AUTO_CMD12_ERR_STATUS_0

Offset: 0x3c I Read/Write: R/W I Reset: 0x000000XX (0b0000xxx000000000xxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
18:16	RW	0x0	UHS_MODE_SEL: UHS Mode Select This field is used to select one of UHS-I modes and effective when 1.8V Signaling Enable is set to 1. If Preset Value Enable in the Host Control 2 register is set to 1, Host Controller sets SDCLK Frequency Select, Clock Generator Select in the Clock Control register and Driver Strength Select according to Preset Value registers. In this case, one of preset value registers is selected by this field. Host Driver needs to reset SD Clock Enable before changing this field to avoid generating clock glitch. After setting this field, Host Driver sets SD Clock Enable again. 000b: SDR12/eMMC LEGACY SPEED 001b: SDR25/eMMC HIGH SPEED SDR 010b: SDR50 011b: SDR104 (SD/SDIO)/HS200 (eMMC) 100b: DDR50 (SD/SDIO)/HIGH SPEED DDR (eMMC) 101b: HS400 (eMMC) When SDR50, SDR104 or DDR50 is selected for SDIO card, interrupt detection at the block gap shall not be used. Read Wait timing is changed for these modes. Refer to the SDIO Specification Version 3.00 for more detail. HS667 is overclocked HS400 mode; Note that tuning should be done in HS200 (reg value=SDR104) mode before entering HS400 or HS667 modes. For HS400 mode: tuning should be in HS200 - SDR at 200 MHz For HS667 mode: tuning should be in HS200 - SDR at 333 MHz - overclocking.

32.9.1.5 Force Event Register

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxxxxxx0x000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR_STATUS 0 = No effect 1 = Force a BOOT_ACK_TIMEOUT_ERR event 2 = Force a BOOT_ACK_ERR event. 3 = Force both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR events.

32.9.2 SDMMC1 Vendor Registers

32.9.2.1 SDMMCA_VENDOR_CLOCK_CNTRL_0

The following registers are Vendor Specific Registers and are mapped to Vendor Specific Address Space (0x100 - 0x1FF)

Vendor Clock Control Register

Offset: 0x100 I Read/Write: R/W I Reset: 0x0000d02d (0bxx00000000000000011010000x0101101)

Bit	Reset	Description
29	0x0	DIFF_CLK_SEL: If set, selects differential CLK and DQS; Used by eMMC IOBRICK only. Software should set this appropriately based on eMMC part used. E_INPUT_* (CMD/DAT/CLK) of eMMC IOBRICK should be set to 0 when differential signaling is used. default is '0' - selects single ended signaling for clk/dqs
28:24	0x0	TRIM_VAL: Tap value for output data path trimmer This determines the trimmer value needed to drive the output data correctly. The tap for outbound trimmer is single MUX. The trim settings required are within very small (0-3) with absolute delay requirement of ~400 ps. Minimal change with PVT variations. Following values are recommended. These are common for all speed modes of SD/eMMC. These are the initial values to start with. SDMMC1 - 2 SDMMC2 - 8 SDMMC3 - 3 SDMMC4 - 8
23:16	0x0	TAP_VAL: Tap value for input data path trimmer This determines the tap value needed to sample the input data correctly. Delay per each tap can range from 70 ps (hv_ff) to 505 ps (lv_ss). Following values are recommended. These are the initial values to start with. 1> SDR12/SDR25/HIGH SPEED DDR modes: If I/O pad trimmer is used (default): SDMMC1 - 4 SDMMC2 - 0 SDMMC3 - 3 SDMMC4 - 0 If core legacy trimmer is used (not recommended): SDMMC1 - 3 SDMMC2 - 3 SDMMC3 - 3 SDMMC4 - 3 2> SDR50 and SDR104/HS200/HS400/HS667 should use tuning procedure to determine the tap value
15:8	0xd0	BASE_CLK_FREQ: System software programs the base SD/MMC clock frequency into this register before loading the SD/MMC driver.This should not be touched when software wants to use SD card in uhs-II mode. This value is readable in the standard, but not writable, SDMMC_CAPABILITIES_0_BASE_CLOCK_FREQUENCY register field. Software should program the actual clock frequency programmed in CAR registers CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC*_0 for each SDMMC controller. This should be done after every time SDMMC is reset and after every soft reset. This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.

Bit	Reset	Description
6	0x0	LEGACY_CLKEN_OVERRIDE: Override for sdmmc_legacy_g_clk clken; Set this to 0 when in UHS-II mode to save power 0 = NORMAL: 0 -> sdmmc_legacy_g_clk is gated 1 = OVERRIDE: 1 -> sdmmc_legacy_g_clk is not gated
5	0x1	SDR50_TUNING_OVERRIDE: override the SDR50_TUNING capabilities bit. Software should only set this bit if it is required to use Tuning for SDR50. (only supported for SDMMC1 and SDMMC3) 0 = NORMAL: 0 -> No Tuning support advertised for SDR50 mode. 1 = OVERRIDE: 1 -> Tuning support is enabled for SDR50 mode.
4	0x0	UHS2_CAPABILITY_OVERRIDE: override the UHS-II capabilities bit. 0 = NORMAL: 0 -> UHS-II support is unadvertised 1 = OVERRIDE: 1 -> UHS-II support is advertised
3	0x1	PADPIPE_CLKEN_OVERRIDE: Override for padmacro and pipemacro clken. 0 = NORMAL: 0 -> CLKEN is de-asserted when internal CLKEN is de-asserted. 1 = OVERRIDE: 1 -> CLKEN is kept asserted even when internal CLKEN is de-asserted. Software should always program this to 0x1 (OVERRIDE).
2	0x1	SPI_MODE_CLKEN_OVERRIDE: Override for CLKEN during SPI_MODE during sw_reset. 0 = NORMAL: 0 -> CLKEN is de-asserted while doing sw_reset. 1 = OVERRIDE: 1 -> CLKEN is kept asserted while doing sw_reset. Software should always program this to 0 (NORMAL).
1	0x0	INPUT_IO_CLK: Feedback clock is selected by default. Software should not change this. Disabling Feedback clock will select Internal Clock that requires different tap value programming. 0 = FEEDBACK 1 = INTERNAL
0	0x1	SDMMC_CLK: This is set when sdmmc_clk is supplied by the CAR module.Prior to sdmmc_clk switch OFF.This bit should be written '0'. By writing zero, the asynchronous card interrupt is routed to the Interrupt controller. 0 = DISABLE 1 = ENABLE

32.9.2.2 SDMMCA_VENDOR_SYS_SW_CNTRL_0

Vendor System Software Control Register

Offset: 0x104 I Read/Write: R/W I Reset: 0x38600002 (0b00111000011xxxxxx0000000x0000010)

Bit	Reset	Description
31	0x0	ENHANCED_STROBE_MODE: Enables enhanced strobe mode in HS400/HS667 mode for eMMC5.x devices 0 - cmd_in(Resp) is sampled by loopback clock which requires tuning 1 - cmd_in(Resp) is sampled by DQS_in which requires no tuning software has to set this bit appropriately based on device capability since this is an optional feature for eMMC5.x devices. If device supports this feature, software should set this bit to avoid tuning.
30	0x0	USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT: When set, uses TMCLK data timeout counter for generating wr_crc_status data-timeout When cleared, uses sdmmc_clk for maintaining wr_crc_status data timeout counter
29	0x1	USE_NCRC_FOR_WR_CRC_STATUS_TIMEOUT_VAL:This field is valid only when USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT is set to 0. When cleared, uses data timeout value as wr CRC status timeout value (specification defined one) When set, uses Ncrc cycles as timeout value
28	0x1	USE_TMCLK_FOR_DATA_TIMEOUT: When set, uses TMCLK data timeout counter for generating legacy data timeout error (except wr_crc_status timeout) When cleared, uses sdmmc_clk for maintaining data timeout counter
27:24	0x8	DEVICE_BUSY_WAIT_CYCLES:In HS400/667 mode, busy should be sampled using clock instead DQS.So, Dat0 from card is passed to two different async FIFOs (one is running in sdmmc_clk and other is in sdmmc_dqs).CRC status is taken from dat0 sampled in DQS and busy should be checked after CRC status end bit reception using dat0 sampled in clk.Both clk and DQS dat0 paths don't have same propagation delay due to sync-ers used in async FIFOs and also due to phase difference between clk and dqs.Due to this delay difference between the paths, we may sample wrong busy status if we sample busy immediately after receiving END bit of CRC status. To avoid incorrect sampling of busy, a wait state before busy polling is added. This register field is used to load wait_cycles counter. Note that this counter is used only in HS400/667 mode.
23	0x0	ALLOW_CARD_CLK_STALLS_IN_WR: When set, allows card clock stopping during transfer of data within a block in HIGH SPEED DDR/HS400 writes.
22	0x1	EMMC_IOBRICK_CLK_DATA:Used to drive AP_CLK and AN_CLK input of iobrick.0x1 - clk_out will be same as iobrick_clk_in0x0 - clk_out will be inverted iobrick_clk_in
21	0x1	QUALIFY_WITH_RD_DATA_VLD:We have async FIFOs in both cmd_in and dat_in paths in padmacro which are used in tunable modes.When this bit set, rdata from FIFO is treated as valid data only when rd_req is high. This is needed to handle bubbles on 'rd_req' when MTBF is high.
14	0x0	SD_BUS_POWER_ON_OFF_INT_STATUS: SD_BUS_POWER was changed. System software can use this interrupt to implement power switch.

Bit	Reset	Description
13	0x0	VOLT_SWITCH_INT_STATUS: VOLT_18_EN was changed. System software can use this interrupt to implement a UHS-I voltage switch procedure for a standard SD Host driver 0 = NO_INT 1 = GEN_INT
12	0x0	TUNING_SYS_INT_STATUS: CMD19 was issued while EXECUTE_TUNING was set. System software can use this interrupt to implement a UHS-I tuning procedure for a standard SD Host driver. 0 = NO_INT 1 = GEN_INT
11:8	0x0	TUNING_ASYNC_FIFO_ADDNL_DELAY:In tunable modes, data/cmd IB path uses async FIFO which contributes additional delay to cmd/resp and wdata/CRC token round trip delay.Controller waits for this round trip delay before waiting for Ncr or Ncrc. This is required not to latch previous data/cmd driven by host itself as start bits of resp or CRC.Round trip delay is calculated for async FIFO with sync-2d.This register field holds the additional delay in cycles which should be added to round trip delay. This additional delay is caused by increasing sync-er depth (more than 2) or addition of output flop stage in async FIFO. Default value is zero. NOTE: Software should not update this field unless a new PROD setting is given.
6	0x0	SD_BUS_POWER_ON_OFF_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when SD_BUS_POWER is changed. 0 = DISABLE 1 = ENABLE
5	0x0	VOLT_SWITCH_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when VOLT_18_EN is changed. 0 = DISABLE 1 = ENABLE
4	0x0	TUNING_SYS_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when CMD19 is issued while EXECUTE_TUNING is set. 0 = DISABLE 1 = ENABLE
3	0x0	ASSERT_BUFF_RD_RDY_INT: Write a 1 to this field to assert sdmmc_interrupt_status_0_buffer_read_ready. Used by the system software that implements the tuning procedure to signal to the standard SD driver that the tuning process has completed 0 = DISABLE 1 = ENABLE
2	0x0	IO_TRIM_BYPASS:Override bit for selecting between core trimmer (Vcore dependent) and I/O trimmer (custom trimmer) in IB clock path. Default option is I/O trimmer; software should not set this field.
1	0x1	INT_MASK_WHILE_TUNING: As per specification, Host should not generate any interrupts (including cmd_complete and data_xfer_complete) except buffer_read_ready interrupt during tuning sequence is being performed software can override this behavior by clearing this bit - but this leads to a specification violation 0 = DISABLE 1 = ENABLE
0	0x0	SPI_MODE: This is a mirror bit.The SPI mode is set if this bit is set or CMD_XFER_MODE[7] is set Writing 1 will drive the CS Low and writing zero will de-assert the CS Signal 0 = DISABLE : 0 -> SPI mode is disabled. 1 = ENABLE : 1 -> SPI mode is enabled.

32.9.2.3 SDMMCA_VENDOR_ERR_INTR_STATUS_0

Legacy Interrupt Status Register

The fields are valid when an error interrupt has occurred.

Offset: 0x108 I Read/Write: RO I Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	SDMMC_LEGACY_CTLR_IDLE: indicates legacy SD interface controller is idle - no active data transfers on legacy SD interface
17	X	READ_DATA_TIMEOUT: valid when a data timeout error occurs
16	X	WRITE_CRC_STATUS_TIMEOUT: valid when a data timeout error occurs
15	X	WRITE_BUSY_TIMEOUT: valid when a data timeout error occurs
14	X	RESP_BUSY_TIMEOUT: valid when a data timeout error occurs
13	X	SPI_WRITE_BUSY_TIMEOUT
12	X	SPI_RX_START_TOKEN_TIMEOUT
8:5	X	SPI_DAT_ERR_TOKEN: Data Error Token,while read from card.

Bit	Reset	Description
4:0	X	SPI_DAT_RESPONSE: Data Response while write to card 5 = DATA_ACCEPTED 11 = CRC_ERR 13 = WRITE_ERR

32.9.2.4 SDMMCA_VENDOR_CAP_OVERRIDES_0

Capabilities Override Bits

Offset: 0x10c I Read/Write: R/W I Reset: 0x00000007 (0bxxx0xxxxxxxxxxxxx000000xxxx0111)

Bit	Reset	Description
28	0x0	CMD_QUEUING_MODE_EN: useful for eMMC5.x devices which support CMD queuing. Software should enable this when it wants to use cmd queuing of eMMC5.x devices in HS400 mode.
13:8	0x0	DQS_TRIM_VAL: PROD_VAL=0x11 - Tap value for incoming DQS path trimmer - used in HS400/HS667 modes
3	0x0	DRV_LPBK_CLK_ON_CMD_LINE: Loopback trimmed clock will be driven onto cmd line, if this bit set to 1. Should be set to zero during normal data transfers.Useful in debug.
2	0x1	VOLTAGE_3_3_V_SUPPORT_OVERRIDE: Voltage support 3_3_V override
1	0x1	VOLTAGE_3_0_V_SUPPORT_OVERRIDE: Voltage support 3_0_V override
0	0x1	VOLTAGE_1_8_V_SUPPORT_OVERRIDE: Voltage support 1_8_V override

32.9.2.5 SDMMCA_VENDOR_BOOT_CNTRL_0

Vendor Boot Control Register

This register is used to configure Boot Mode to support MMC v4.3 cards.

Offset: 0x110 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BOOT_ACK: This bit is used to support Boot Option in MMC 4.3 version cards. If set Boot acknowledgment is given by card else not given by card 0 = DISABLE 1 = ENABLE
0	0x0	BOOT: This bit enables/disable BootOption1.If set BootOption1 is enabled, hardware auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

32.9.2.6 SDMMCA_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 0x114 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles. Boot Acknowledgment Timeout error occurs (VENDOR_SPECIFIC_ERR[0])

32.9.2.7 SDMMCA_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 0x118 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles. Then Data Timeout error occurs.

32.9.2.8 SDMMCA_VENDOR_DEBOUNCE_COUNT_0

Debounce Counter Value Register

The Debounce Counter runs on 32KHz clock. Keeping the default value to 100 ms = (100 * 32cycles/1 ms) = 3200 cycles for 100 ms = 0xC80

Offset: 0x11c I Read/Write: R/W I Reset: 0x00000c80 (0bxxxxxxxx00000000000110010000000)

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32KHz clock cycles is programmed to meet Debounce period of the card slot.

32.9.2.9 SDMMCA_VENDOR_MISC_CNTRL_0

Miscellaneous Vendor Control Register

SDMMC_SPARE0: Spare register bits with reset value of 0

SDMMC_SPARE0[0]: SW_RESET_CLKEN_OVERRIDE, override the sdmmc_clken when doing SW_RESET if set to 1.

SDMMC_SPARE0[1]: When set, allows SD clock to be stopped in the middle of a read data block while in SDR104/ HS400/HS667 modes(allow_sdr104_intrablock_stalls).

Unsafe for at least some SD/eMMC cards, but may improve SDR104 DMA read performance in some cases.

SDMMC_SPARE0[2]: When set, SDR104 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR104

SDMMC_SPARE0[3]: When set, SDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR50

SDMMC_SPARE0[5]: When 0, masks the pad macro's "high speed" enable to 0, causing the pad macro to always launch data on the falling edge of the clock. This prevents the SD Host driver's setting of SDMMC_POWER_CONTROL_HOST_x_HIGH_SPEED_EN from undesirably affecting the output timing.

SDMMC_SPARE0[7:6]: Number of pipe stages.

SDMMC_SPARE0[8]: When set, DDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_DDR50

SDMMC_SPARE1: Spare register bits with reset value of 1

SDMMC_SPARE1[0]: CMD_CONFLICT_DISABLE, disable command line conflict if set to 1.

SDMMC_SPARE1[1]: Control bit to select between internal and external loop back clock. 1-external 0-internal

Tegra X1 does not support external loopback for any SDMMC controller.

Offset: 0x120 I Read/Write: R/W I Reset: 0xffff0098 (0b11111111111111110000000010011000)

Bit	Reset	Description
31:16	0xffff	SDMMC_SPARE1: Spare register bits with reset value of 1
15:1	0x4c	SDMMC_SPARE0: Spare register bits with reset value of 0x40
0	0x0	ERASE_TIMEOUT_LIMIT: Erase timeout value. 0 = FINITE: Finite, It is limited to the programmed value in the DATA_TIMEOUT_VALUE 1 = INFINITE: Infinite,Controller would be monitoring until the card is busy.

32.9.2.10 SDMMCA_MAX_CURRENT_OVERRIDE_0

Offset: 0x124 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	OVERRIDE_FOR_1_8V: Maximum override for 1.8V VDD1
15:8	0x0	OVERRIDE_FOR_3_0V: Maximum override for 3.0V VDD1
7:0	0x0	OVERRIDE_FOR_3_3V: Maximum override for 3.3V VDD1

32.9.2.11 SDMMCA_MAX_CURRENT_OVERRIDE_HI_0

Offset: 0x128 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	OVERRIDE_FOR_1_8V_VDD2: Maximum override for 1.8V VDD2

32.9.2.12 SDMMCA_VENDOR_IO_TRIM_CNTRL_0

Vendor I/O trimmer control register

Used to configure I/O trimmer

Table 207: Truth Table

Input Pins			Output	Comments
E_DPD	SEL_VREG	SEL_VREF	CLKOUT	
1	x[1]	x	0	The cell is in deep power down mode.
0	1	x	based on ip_clk_select selected clock input	Trimmer is powered by VAUXC.
0	0	x	based on ip_clk_select selected clock input	Trimmer is powered by regulated voltage.

1. 'x' indicates don't care.

Offset: 0x1ac I Read/Write: R/W I Reset: 0x00000015 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010101)

Bit	Reset	Description
5	ENABLE	TRIM_PWRSAVE: Enables power saving mode by clock gating the unused taps in delay chain Active low signal, 0 - power saving mode enabled - clock gating is enabled for unused trimmer taps - may affect tap delay 1 - no power saving - all the trimmer taps are not clock gated 0 = ENABLE 1 = DISABLE
4:3	VREF_925_MV	SEL_VREF_LEVEL: Selects Vref voltage level 0 = VREF_875_MV 1 = VREF_900_MV 2 = VREF_925_MV 3 = VREF_950_MV
2	0x1	SEL_VREG: Select voltage supply for delay chain present in both Trimmer and DLLPROD value: 0x0 ***Software should set this to 0x0 before accessing SD/eMMC. This setting makes IB trimmer delay independent of VDD_CORE*** 0 - selects regulated reference voltage for trimmer and DLL supply - default (recommended option for tunable SD/eMMC modes) 1 - selects VAUXC for trimmer and DLL supply and shut down BG+REG circuit (can be used in non-tunable modes for power saving) Power up time for BG+REG is ~3us(worst case). Power down time for BG+REG is ~1 us (worst case). If software wants to turn on/off BG+REG when SDMMC is idle, it has to take hit of 3 us power on time. When SEL_VREG is toggled, both DLL and rx clock trimmer output could glitch irrespective of input clock state which could cause corresponding rx CMD and DATA FIFOs to go into bad state. Software should issue SW_RESET_DAT and SW_RESET_CMD to reset host FIFOs after BG <-> VAUXC switching. This would ensure error free data transfers from there on. Powering down BG would need 3 us turn ON time which may cause IOPS reduction, if software shuts down BG after every transfer and enables it on seeing new transfer request; it may not be possible to do dynamic shut down of BG without stalling new requests.
1	0x0	SEL_VREF: Select reference voltage for voltage regulator 0 - selects Bandgap Voltage Reference (recommended option for SD/eMMC tunable modes) 1 - selects resistor divider voltage reference and power down bandgap Switching time between the supplies is 1 us. When switching from one supply to other supply, we need to wait for at least 1 us before doing any data transfers.
0	0x1	PD_BGREG: Not used - Dummy control Power down Band Gap voltage reference, voltage regulator and resistor chain voltage ref (BG+REG) present in custom I/O trimmer used for SD/eMMC bus tuning. Active High signal, PD_BGREG=1 => Power down BG+REG; PD_BGREG=0 => power up Power down and up time for BG+REG is 1 us. If software wants to turn on/off BG+REG when SDMMC is idle, it has to follow 1us power on/off time. Back-up option for powering down BG. Software should clear this bit when it wants to turn ON BG by setting SEL_VREG=0. The original idea to have PD_BGREG pin is to provide power saving feature when the eMMC/SDMMC is in IDLE state, but not in DPD mode. This pin function is actually merged into SEL_VREG function -BG+REG circuit is shut-down when SEL_VREG=1 (trimmer powered by VAUXC)

32.9.2.13 SDMMCA_VENDOR_TUNING_CNTRL0_0

Vendor Tuning Control0 Register

Offset: 0x1c0 I Read/Write: R/W I Reset: 0x74020090 (0bx111010000000001x0000000010010000) I Default: 0x00000040

Bit	Reset	SW Default	Description
30	0x1	_NONE_	RD_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on read data CRC error
29	0x1	_NONE_	WR_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on write CRC error
28	0x1	_NONE_	CMD_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on cmd CRC error
27	0x0	_NONE_	RETUNING_REQ_EN_ON_CRC_ERR_DETECTION: Re-tuning request is generated when set to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
26	0x1	_NONE_	TUNING_ERR_EN_ON_CRC_ERR_DETECTION: Tuning error is generated to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
25:18	0x0	_NONE_	START_TAP_VAL:start tap value to be used by tuning; start_tap should be multiple of step_size chosen and its valid range is 0-255; Not valid when TAP_VAL_UPDATED_BY_HW is set to zero. Tuning algorithm uses this as the start tap value for scanning through trimmer taps.
17	0x1	_NONE_	TAP_VAL_UPDATED_BY_HW: This bit is functional only in tunable modes (SDR50, SDR104 and HS200). Software can choose to update the tap value by itself by clearing this bit; Preferred value is 1 - tap value is updated by hardware. If this bit is cleared, hardware does not update tap_val per every tuning iteration. Software can update it as desired. Tuning pattern match is indicated by sampling_clock_select per every tuning iteration. Software has to maintain the status of each tuning iteration and determine the best PASS window to fix the final sampling point. Software can program NUM_TUNING_ITERATIONS as desired. Once the number of tuning commands issued reaches number of tuning iterations programmed, execute_tuning bit will be cleared to indicate the completion of tuning procedure. Note that using this option violates Host specification but provided for legacy reasons. It helps us in using legacy software tuning solution in case hardware solution does not work.
15:13	TRIES_40	_NONE_	NUM_TUNING_ITERATIONS: The number of tuning iterations to be used by tuning circuit; Preferred value is 40 iterations as per Host specification. Using other values violates the specification but provided for trimmer characterization purpose. 0 = TRIES_40 1 = TRIES_64 2 = TRIES_128 3 = TRIES_192 4 = TRIES_256
12:6	0x2	0x1	MUL_M: implements a multiplier - M+1 Final tap value is derived from best passing window and calculated as follows. Final tap value = first_pass + ((last_pass - first_pass)*Q); where Q = percentage of pass window;default-75% Q = M+1/(2^N); N:1...7 M:should be in range [0:2^N-1];
5:3	0x2	_NONE_	DIV_N: implements a divider - 2^N; maximum div is 2^7 =>128
2:0	0x0	_NONE_	TUNING_WORD_SEL: Selects desired word from 256-bit tuning status bitmap status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.2.14 SDMMCA_VENDOR_TUNING_CNTRL1_0

Vendor Tuning Control1 register

Different step size is required in SDR50 mode to cover two UI (100 MHz => 2*10 ns)

With 70 ps/tap trimmer resolution, we can cover almost 2 UI using step_size=8 in SDR50 and step_size=4 in SDR104.

Tuning will be done in HS200 - SDR mode only.

HS200 - tuning at 200 MHz - UHS_SEL should be SDR104 for executing tuning

Before initiating data transfers in HS400 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 200 MHz

For overclocked modes: Before initiating data transfers in HS667 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 333 MHz

Note: *eMMC bus tuning will always be run in HS200 mode only.*

Use step_size = 1 to scan through all taps due to hardware issue.

Offset: 0x1c4 I Read/Write: R/W I Reset: 0x00000023 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010x011) I Default: 0x00000000

Bit	Reset	SW Default	Description
6:4	0x2	0x0	STEP_SIZE_SDR104_HS200: tap_val is incremented by step_size for every tuning iteration - used in SDR104/ HS200/HS400 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD For overclocking modes, set this field to 0x1
2:0	0x3	0x0	STEP_SIZE_SDR50: tap_val is incremented by step_size for every tuning iteration - used in SDR50 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD

32.9.2.15 SDMMCA_VENDOR_TUNING_STATUS0_0

Vendor Tuning Status0 register

Offset: 0x1c8 I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	STATUS_WORD: Each bit indicates the status of each tuning iteration, when tap value is updated by hardware (TAP_VAL_UPDATED_BY_HW=1); 0-Tuning pattern not matched 1-tuning pattern matched We have a total of 256 tap values. Software can issue a maximum.of 256 tuning commands for debug. Software needs to read this register eight times to get status of all 256 iterations by changing tuning_word_sel status[255:0] is left shifted and loaded into this register every time when tuning_word_sel is changed status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.2.16 SDMMCA_VENDOR_TUNING_STATUS1_0

Vendor Tuning Status1 register

Offset: 0x1cc I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	PASS_WINDOW_END_BEFORE_FINE_TUNING: End tap value of best PASS window found by scan FSM
23:16	X	PASS_WINDOW_START_BEFORE_FINE_TUNING: Start tap value of best PASS window found by scan FSM
15:8	X	PASS_WINDOW_END_AFTER_FINE_TUNING: End tap value of best PASS window after fine tuning
7:0	X	PASS_WINDOW_START_AFTER_FINE_TUNING: Start tap value of best PASS window after fine tuning

32.9.2.17 SDMMCA_VENDOR_CLK_GATE_HYSTERESIS_COUNT_0

Vendor Clk gating Hysteresis Counter initial value

Offset: 0x1d0 I Read/Write: R/W I Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx001111)

Bit	Reset	Description
5:0	0xf	CLK_COUNT: Before gating second level clocks controller will wait for these many cycles. we can recover if any idle windows are missed in clken equation

32.9.2.18 SDMMCA_VENDOR_PRESET_VAL0_0

Vendor Preset Value Registers

SD host specification defines one preset value register for each bus speed mode which should be set by host by some unique method.

Preset values vary based on the base frequency used which is in software (Tegra system driver) control.

Tegra System driver supposed to set BASE_CLK_FREQ in VENDOR_CLOCK_CNTRL register before handling over the control to SD host standard driver.

In the similar way, system driver should set below vendor preset values based on the base clock frequency and the desired card clock frequency in each bus speed mode

This should be done after every time SDMMC is reset and after every soft reset.

This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.

Default values are set assuming base clock frequency=208 MHz.

Offset: 0x1d4 I Read/Write: R/W I Reset: 0x00201000 (0bxx00000000100000000100000000000)

Bit	Reset	Description
29:20	0x2	SDCLK_FREQ_SEL_HIGH_SPEED: System software programs 10-bit divider value to generate SDCLK in default speed mode (<50 MHz, 3.3V signaling) This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
19:10	0x4	SDCLK_FREQ_SEL_DEFAULT: System software programs 10-bit divider value to generate SDCLK in default speed mode (<25 MHz, 3.3V signaling). This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x4 assuming 208 MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_INIT: System software programs 10-bit divider value to generate desired SDCLK frequency during initialization. This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_LOW register field. For example, if 400 kHz SDCLK is desired at base clk freq=48 MHz, this register should be programmed with 0x3C

32.9.2.19 SDMMCA_VENDOR_PRESET_VAL1_0

Offset: 0x1d8 I Read/Write: R/W I Reset: 0x00100804 (0bxx00000000100000000100000000100)

Bit	Reset	Description
29:20	0x1	SDCLK_FREQ_SEL_SDR50: System software programs 10-bit divider value to generate SDCLK in SDR50 mode (<100 MHz, 1.8V signaling) This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x1 (gives 2N divider) assuming 208 MHz base clock
19:10	0x2	SDCLK_FREQ_SEL_SDR25: System software programs 10-bit divider value to generate SDCLK in SDR25 mode (<50 MHz, 1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
9:0	0x4	SDCLK_FREQ_SEL_SDR12: System software programs 10-bit divider value to generate SDCLK in SDR12 mode (<25 MHz, 1.8V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_HIGH register field

32.9.2.20 SDMMCA_VENDOR_PRESET_VAL2_0

Offset: 0x1dc I Read/Write: R/W I Reset: 0x00000800 (0bxxxxxxxxxxxxxx00000000100000000000)

Bit	Reset	Description
19:10	0x2	SDCLK_FREQ_SEL_DDR50: System software programs 10-bit divider value to generate SDCLK in DDR50 mode (<50MHz,1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x2 assuming 208MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_SDR104: System software programs 10-bit divider value to generate SDCLK in SDR104 mode (<208MHz,1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_LOW register field

32.9.2.21 SDMMCA_SDMEMCOMPPADCTRL_0

SDMEMCOMP Pad control register

This register is used to control COMP pad inputs.

Offset: 0x1e0 I Read/Write: R/W I Reset: 0x88000001 (0b10x01xxxxxxxxxxxxxxxx0xxx0000001) I Default: 0x00000000

Bit	Reset	SW Default	Description
31	0x1	0x0	PAD_E_INPUT_OR_E_PWRD: used to control E_INPUT (for SDMMC1/3) and E_PWRD (for SDMMC2/4) input of pu/pd comp pad should be set before starting auto-cal and cleared once auto-calibration is done (for power saving) NOTE: E_PWRD = !PAD_E_INPUT_OR_E_PWRD and E_INPUT = PAD_E_INPUT_OR_E_PWRD
30	0x0	_NONE_	COMP_PAD_REG_ON: If set, turns ON regulator in comp pad. Software should not set this bit.
28:27	0x1	_NONE_	COMP_PAD_DRV_TYPE: used to control drv_type input of BDSDMEMLVCOMP_C pad
10	0x0	_NONE_	COMP_PAD_E_TEST_OUT: used to control e_test_out input of COMP pad
6:4	0x0	_NONE_	COMP_PAD_TEST_SEL: used to control test_sel input of COMP pad
3:0	0x1	_NONE_	SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL: used to control vref_sel input of COMP pad. Software should set this to 0x7.

32.9.2.22 SDMMCA_AUTO_CAL_CONFIG_0

SDMEMCOMP pad auto-calibration settings

AUTO_CAL_SLW_OVERRIDE

0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output

DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3]

1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs

AUTO_CAL_OVERRIDE

0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting

1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly

Offset: 0x1e4 I Read/Write: R/W I Reset: 0x00010000 (0b0000xxxxxxxxx001x0000000x0000000) I Default: 0x20000000

Bit	Reset	SW Default	Description
31	0x0	_NONE_	AUTO_CAL_START: Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	0x0	_NONE_	AUTO_CAL_OVERRIDE: AUTOCAL override. 0 = NORMAL: 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 = OVERRIDE: 1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly
29	DISABLED	0x1	AUTO_CAL_ENABLE: AUTOCAL enable. 0 = DISABLED: 0 (disabled): use sdmmc2tmc_cfg* register settings for pullup/dn 1 = ENABLED: 1 (normal operation): use SDMMC generated pullup/dn (override or AUTOCAL)
28	0x0	_NONE_	AUTO_CAL_SLW_OVERRIDE: AUTOCAL slew rate override 0 = NORMAL: 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3] 1 = OVERRIDE: 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs
18:16	0x1	_NONE_	AUTO_CAL_STEP: calibration step interval (in microseconds)
14:8	0x0	_NONE_	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
6:0	0x0	_NONE_	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

32.9.2.23 SDMMCA_AUTO_CAL_INTERVAL_0

SDMEMCOMP pad calibration interval - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1e8 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	AUTO_CAL_INTERVAL: 0: do calibration once Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.This feature is not supported from Tegra X1 due to movement of level shifters to COMP pad. Software should trigger calibration at regular intervals, if needed.

32.9.2.24 SDMMCA_AUTO_CAL_STATUS_0

SDMEMCOMP pad calibration status - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1ec I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (AUTO_CAL_ACTIVE == 0)
30:24	X	AUTO_CAL_PULLDOWN_ADJ: Pulldown code sent to pads
22:16	X	AUTO_CAL_PULLUP_ADJ: Pullup code sent to pads
14:8	X	AUTO_CAL_PULLDOWN: Pulldown code generated by auto-calibration
6:0	X	AUTO_CAL_PULLUP: Pullup code generated by auto-calibration

32.9.2.25 SDMMCA_IO_SPARE_0

SPARE bits. These SPARE_OUT bits go to pipe -> pad and then come back as SPARE_IN

SPARE_OUT[3]: IO_SPARE[19] - used as MUX select which selects between one cycle delay and two cycle delay versions of cmd_oen to mask wdata of IB capture flop.

0x0: selects two cycle delayed version

0x1: selects one cycle delayed version - recommended

SPARE_OUT[2]: IO_SPARE[18] - used as active low enable for gating both CMD_IN and DAT_IN async FIFOs wdata when we are driving CMD/DAT lines.

0x0: write 1 when OEN is active and Zi value when OEN is not active into FIFO (default)

0x1: write Zi value into FIFO irrespective of OEN state.

Offset: 0x1f0 I Read/Write: R/W I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx) I Default: 0x00080008

Bit	R/W	Reset	SW Default	Description
31:16	RW	X	0x8	SPARE_OUT:PROD value: Software should set SPARE_OUT[3]: IO_SPARE[19] to 1 before accessing SD/eMMC.
15:0	RO	X	0x8	SPARE_IN

32.9.2.26 SDMMCA_SDMMCA_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register.

> ***Note:*** *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility*

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clk is enabled.

With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC.

A '1' written to the CCLK override field keeps client's clk always on inside MCCIF.

Offset: 0x1f4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	0x0	SDMMCA_RCLK_OVR_MODE
19	0x0	SDMMCA_WCLK_OVR_MODE
18	0x0	SDMMCA_CCLK_OVERRIDE
17	0x0	SDMMCA_RCLK_OVERRIDE
16	0x0	SDMMCA_WCLK_OVERRIDE

32.9.2.27 SDMMCA_TIMEOUT_WCOAL_SDMMCA_0

Write Coalescing Time-Out Register

> ***Note:*** *Write coalescing is no longer supported by the MCCIF clients.*

Registers are retained for software compatibility but are not used by the hardware.

Offset: 0x1f8 I Read/Write: R/W I Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	SDMMCWA_WCOAL_TMVAL

32.9.3 SDMMC2 Vendor Registers

32.9.3.1 SDMMCAA_VENDOR_CLOCK_CNTRL_0

The following registers are Vendor Specific Registers and are mapped to Vendor Specific Address Space (0x100 - 0x1FF)

Vendor Clock Control Register

Offset: 0x100 I Read/Write: R/W I Reset: 0x0000d02d (0bxx00000000000011010000x0101101)

Bit	Reset	Description
29	0x0	DIFF_CLK_SEL: If set, selects differential CLK and DQS; Used by eMMC IOBRICK only. Software should set this appropriately based on eMMC part used. E_INPUT_* (CMD/DAT/CLK) of EMMC IOBRICK should be set to 0 when differential signaling is used. default is '0' - selects single ended signaling for clk/dqs
28:24	0x0	TRIM_VAL: Tap value for output data path trimmer This determines the trimmer value needed to drive the output data correctly. The tap for outbound trimmer is single MUX. The trim settings required are within very small (0-3) with absolute delay requirement of ~400 ps. Minimal change with PVT variations. Following values are recommended. These are common for all speed modes of SD/eMMC. These are the initial values to start with. SDMMC1 - 2 SDMMC2 - 8 SDMMC3 - 3 SDMMC4 - 8

Bit	Reset	Description
23:16	0x0	TAP_VAL: Tap value for input data path trimmer This determines the tap value needed to sample the input data correctly. Delay per each tap can range from 70ps (hv_ff) to 505ps (lv_ss). Following values are recommended. These are the initial values to start with. 1> SDR12/SDR25/High Speed DDR modes: If I/O pad trimmer is used (default): SDMMC1 - 4 SDMMC2 - 0 SDMMC3 - 3 SDMMC4 - 0 If core legacy trimmer is used: SDMMC1 - 3 SDMMC2 - 3 SDMMC3 - 3 SDMMC4 - 3 2> SDR50 and SDR104/HS200/HS400/HS667 should use tuning procedure to determine the tap value
15:8	0xd0	BASE_CLK_FREQ: System software programs the base SD/MMC clock frequency into this register before loading the SD/MMC driver.This should not be touched when software wants to use SD card in uhs-II mode. This value is readable in the standard, but not writeable, SDMMC_CAPABILITIES_0_BASE_CLOCK_FREQUENCY register field. Software should program the actual clock frequency programmed in CAR registers CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC*_0 for each SDMMC controller. This should be done after every time SDMMC is reset and after every soft reset. This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.
6	0x0	LEGACY_CLKEN_OVERRIDE: Override for sdmmc_legacy_g_clk clken; Set this to 0 when in UHS-II mode to save power 0 = NORMAL: 0 -> sdmmc_legacy_g_clk is gated 1 = OVERRIDE: 1 -> sdmmc_legacy_g_clk is not gated
5	0x1	SDR50_TUNING_OVERRIDE: override the SDR50_TUNING capabilities bit. Software should only set this bit if it is required to use Tuning for SDR50. (only supported for SDMMC1 and SDMMC3) 0 = NORMAL: 0 -> No Tuning support advertised for SDR50 mode. 1 = OVERRIDE: 1 -> Tuning support is enabled for SDR50 mode.
4	0x0	UHS2_CAPABILITY_OVERRIDE: override the UHS-II capabilities bit. 0 = NORMAL: 0 -> UHS-II support is unadvertised 1 = OVERRIDE: 1 -> UHS-II support is advertised
3	0x1	PADPIPE_CLKEN_OVERRIDE: Override for padmacro and pipemacro clken. 0 = NORMAL: 0 -> CLKEN is de-asserted when internal CLKEN is de-asserted. 1 = OVERRIDE: 1 -> CLKEN is kept asserted even when internal CLKEN is de-asserted. Software should always program this to 0x1 (OVERRIDE).
2	0x1	SPI_MODE_CLKEN_OVERRIDE: Override for CLKEN during SPI_MODE during sw_reset. 0 = NORMAL: 0 -> CLKEN is de-asserted while doing sw_reset. 1 = OVERRIDE: 1 -> CLKEN is kept asserted while doing sw_reset. Software should always program this to 0 (NORMAL).
1	0x0	INPUT_IO_CLK: Feedback clock is selected by default. Software should not change this. Disabling Feedback clock will select Internal Clock that requires different TAP Value Programming. 0 = FEEDBACK 1 = INTERNAL
0	0x1	SDMMC_CLK: This is set when sdmmc_clk is supplied by the CAR module.Prior to sdmmc_clk switch OFF.This bit should be written '0'. By writing zero,the asynchronous card interrupt is routed to the Interrupt controller. 0 = DISABLE 1 = ENABLE

32.9.3.2 SDMMCAA_VENDOR_SYS_SW_CNTRL_0

Vendor System Software Control Register

Offset: 0x104 I Read/Write: R/W I Reset: 0x38600002 (0b00111000011xxxxxx0000000x0000010)

Bit	Reset	Description
31	0x0	ENHANCED_STROBE_MODE: Enables enhanced strobe mode in HS400/HS667 mode for eMMC5.x devices 0 - cmd_in(Resp) is sampled by loopback clock which requires tuning 1 - cmd_in(Resp) is sampled by DQS_in which requires no tuning software has to set this bit appropriately based on device capability since this is an optional feature for eMMC5.x devices. If device supports this feature, software should set this bit to avoid tuning.
30	0x0	USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT: When set, uses TMCLK data timeout counter for generating wr_crc_status data-timeout When cleared, uses sdmmc_clk for maintaining wr_crc_status data timeout counter
29	0x1	USE_NCRC_FOR_WR_CRC_STATUS_TIMEOUT_VAL: This field is valid only when USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT is set to 0. When cleared, uses data timeout value as wr CRC status timeout value (specification defined one) When set, uses Ncrc cycles as timeout value
28	0x1	USE_TMCLK_FOR_DATA_TIMEOUT: When set, uses TMCLK data timeout counter for generating legacy data timeout error (except wr_crc_status timeout) When cleared, uses sdmmc_clk for maintaining data timeout counter
27:24	0x8	DEVICE_BUSY_WAIT_CYCLES: In HS400/667 mode, busy should be sampled using clock instead DQS.So, Dat0 from card is passed to two different async FIFOs (one is running in sdmmc_clk and other is in sdmmc_dqs).CRC status is taken from dat0 sampled in DQS and busy should be checked after CRC status end bit reception using dat0 sampled in clk.Both clk and DQS dat0 paths don't have same prop delay due to sync-ers used in async FIFOs and also due to phase difference between clk and dqs.Due to this delay difference between the paths, we may sample wrong busy status if we sample busy immediately after receiving END bit of CRC status. To avoid incorrect sampling of busy, a wait state before busy polling is added. This register field is used to load wait_cycles counter. Note that this counter is used only in HS400/667 mode.
23	0x0	ALLOW_CARD_CLK_STALLS_IN_WR: When set, allows card clock stopping during transfer of data within a block in HIGH SPEED DDR/HS400 writes.
22	0x1	EMMC_IOBRICK_CLK_DATA: Used to drive AP_CLK and AN_CLK input of iobrick.0x1 - clk_out will be same as iobrick_clk_in0x0 - clk_out will be inverted iobrick_clk_in
21	0x1	QUALIFY_WITH_RD_DATA_VLD: We have async FIFOs in both cmd_in and dat_in paths in padmacro which are used in tunable modes.When this bit set, rdata from FIFO is treated as valid data only when rd_req is high. This is needed to handle bubbles on 'rd_req' when MTBF is high.
14	0x0	SD_BUS_POWER_ON_OFF_INT_STATUS: SD_BUS_POWER was changed. System software can use this interrupt to implement power switch.
13	0x0	VOLT_SWITCH_INT_STATUS: VOLT_18_EN was changed. System software can use this interrupt to implement a UHS-I voltage switch procedure for a standard SD Host driver 0 = NO_INT 1 = GEN_INT
12	0x0	TUNING_SYS_INT_STATUS: CMD19 was issued while EXECUTE_TUNING was set. System software can use this interrupt to implement a UHS-I tuning procedure for a standard SD Host driver. 0 = NO_INT 1 = GEN_INT
11:8	0x0	TUNING_ASYNC_FIFO_ADDNL_DELAY: In tunable modes, data/cmd IB path uses async FIFO which contributes additional delay to cmd/resp and wdata/CRC token round trip delay.Controller waits for this round trip delay before waiting for Ncr or Ncrc. This is required not to latch previous data/cmd driven by host itself as start bits of resp or CRC.Round trip delay is calculated for async FIFO with sync-2d.This register field holds the additional delay in cycles which should be added to round trip delay. This additional delay is caused by increasing sync-er depth (more than 2) or addition of output flop stage in async FIFO. Default value is zero. NOTE: Software should not update this field unless a new PROD setting is given.
6	0x0	SD_BUS_POWER_ON_OFF_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when SD_BUS_POWER is changed. 0 = DISABLE 1 = ENABLE
5	0x0	VOLT_SWITCH_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when VOLT_18_EN is changed. 0 = DISABLE 1 = ENABLE
4	0x0	TUNING_SYS_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when CMD19 is issued while EXECUTE_TUNING is set. 0 = DISABLE 1 = ENABLE
3	0x0	ASSERT_BUFF_RD_RDY_INT: Write a 1 to this field to assert sdmmc_interrupt_status_0_buffer_read_ready. Used by the system software that implements the tuning procedure to signal to the standard SD driver that the tuning process has completed 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	IO_TRIM_BYPASS: Override bit for selecting between core trimmer (Vcore dependent) and I/O trimmer (custom trimmer) in IB clock path. Default option is I/O trimmer; software should not set this field.
1	0x1	INT_MASK_WHILE_TUNING: As per specification, Host should not generate any interrupts (including cmd_complete and data_xfer_complete) except buffer_read_ready interrupt during tuning sequence is being performed. Software can override this behavior by clearing this bit - but this leads to a specification violation 0 = DISABLE 1 = ENABLE
0	0x0	SPI_MODE: This is a mirror bit.The SPI mode is set if this bit is set or CMD_XFER_MODE[7] is set Writing 1 will drive the CS Low and writing zero will de-assert the CS Signal 0 = DISABLE: 0 -> SPI mode is disabled. 1 = ENABLE: 1 -> SPI mode is enabled.

32.9.3.3 SDMMCAA_VENDOR_ERR_INTR_STATUS_0

Legacy Interrupt Status Register

The fields are valid when an error interrupt has occurred.

Offset: 0x108 I Read/Write: RO I Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	SDMMC_LEGACY_CTLR_IDLE: indicates legacy SD interface controller is idle - no active data transfers on legacy SD interface
17	X	READ_DATA_TIMEOUT: valid when a data timeout error occurs
16	X	WRITE_CRC_STATUS_TIMEOUT: valid when a data timeout error occurs
15	X	WRITE_BUSY_TIMEOUT: valid when a data timeout error occurs
14	X	RESP_BUSY_TIMEOUT: valid when a data timeout error occurs
13	X	SPI_WRITE_BUSY_TIMEOUT
12	X	SPI_RX_START_TOKEN_TIMEOUT
8:5	X	SPI_DAT_ERR_TOKEN: Data Error Token, while read from card.
4:0	X	SPI_DAT_RESPONSE: Data Response while write to card 5 = DATA_ACCEPTED 11 = CRC_ERR 13 = WRITE_ERR

32.9.3.4 SDMMCAA_VENDOR_CAP_OVERRIDES_0

Capabilities override bits

Offset: 0x10c I Read/Write: R/W I Reset: 0x00000007 (0bxxx0xxxxxxxxxxxxxx000000xxxx0111)

Bit	Reset	Description
28	0x0	CMD_QUEUING_MODE_EN: useful for eMMC5.x devices which support CMD queuing. Software should enable this when it wants to use cmd queuing of eMMC5.x devices in HS400 mode.
13:8	0x0	DQS_TRIM_VAL:PROD_VAL=0x11 - Tap value for incoming DQS path trimmer - used in HS400/HS667 modes
3	0x0	DRV_LPBK_CLK_ON_CMD_LINE: Loopback trimmed clock will be driven onto cmd line, if this bit set to 1. Should be set to zero during normal data transfers.Useful in debug.
2	0x1	VOLTAGE_3_3_V_SUPPORT_OVERRIDE: Voltage support 3_3_V override
1	0x1	VOLTAGE_3_0_V_SUPPORT_OVERRIDE: Voltage support 3_0_V override
0	0x1	VOLTAGE_1_8_V_SUPPORT_OVERRIDE: Voltage support 1_8_V override

32.9.3.5 SDMMCAA_VENDOR_BOOT_CNTRL_0

Vendor Boot Control Register

This register is used to configure Boot Mode to support MMC v4.3 cards.

Offset: 0x110 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BOOT_ACK: This bit is used to support Boot Option in MMC 4.3 version cards. If set Boot acknowledgment is given by card else not given by card 0 = DISABLE 1 = ENABLE
0	0x0	BOOT: This bit enables/disable BootOption1.If set BootOption1 is enabled, hardware auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

32.9.3.6 SDMMCAA_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 0x114 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles. Boot Acknowledgment Timeout error occurs(VENDOR_SPECIFIC_ERR[0])

32.9.3.7 SDMMCAA_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 0x118 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles. Then Data Timeout error occurs.

32.9.3.8 SDMMCAA_VENDOR_DEBOUNCE_COUNT_0

Debounce Counter Value Register

The Debounce Counter runs on 32 kHz clock. Keeping the default value to 100 ms = (100 * 32 cycles/1 ms) = 3200 cycles for 100 ms = 0xC80

Offset: 0x11c I Read/Write: R/W I Reset: 0x00000c80 (0bxxxxxxxx00000000000110010000000)

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32 kHz clock cycles is programmed to meet Debounce period of the card slot.

32.9.3.9 SDMMCAA_VENDOR_MISC_CNTRL_0

Miscellaneous Vendor Control Register

SDMMC_SPARE0: Spare register bits with reset value of 0

SDMMC_SPARE0[0]: SW_RESET_CLKEN_OVERRIDE, override the sdmmc_clken when doing SW_RESET if set to 1.

SDMMC_SPARE0[1]: When set, allows SD clock to be stopped in the middle of a read data block while in SDR104/ HS400/HS667 modes (allow_sdr104_intrablock_stalls).

Unsafe for at least some SD/eMMC cards, but may improve SDR104 DMA read performance in some cases.

SDMMC_SPARE0[2]: When set, SDR104 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR104

SDMMC_SPARE0[3]: When set, SDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR50

SDMMC_SPARE0[5]: When 0, masks the pad macro's "high speed" enable to 0, causing the pad macro to always launch data on the falling edge of the clock. This prevents the SD Host driver's setting of SDMMC_POWER_CONTROL_HOST_x_HIGH_SPEED_EN from undesirably affecting the output timing.

SDMMC_SPARE0[7:6]: Number of pipe stages.

SDMMC_SPARE0[8]: When set, DDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_DDR50

SDMMC_SPARE1: Spare register bits with reset value of 1

SDMMC_SPARE1[0]: CMD_CONFLICT_DISABLE, disable command line conflict if set to 1.

SDMMC_SPARE1[1]: Control bit to select between internal and external loop back clock. 1-external 0-internal

Tegra X1 does not support external loopback for any SDMMC controller.

Offset: 0x120 I Read/Write: R/W I Reset: 0xffff0098 (0b11111111111111110000000010011000)

Bit	Reset	Description
31:16	0xffff	SDMMC_SPARE1: Spare register bits with reset value of 1
15:1	0x4c	SDMMC_SPARE0: Spare register bits with reset value of 0x40
0	0x0	ERASE_TIMEOUT_LIMIT: Erase timeout value. 0 = FINITE: Finite, It is limited to the programmed value in the DATA_TIMEOUT_VALUE 1 = INFINITE: Infinite, Controller would be monitoring until the card is busy.

32.9.3.10 SDMMCAA_MAX_CURRENT_OVERRIDE_0

Offset: 0x124 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	OVERRIDE_FOR_1_8V: Maximum override for 1.8V VDD1
15:8	0x0	OVERRIDE_FOR_3_0V: Maximum override for 3.0V VDD1
7:0	0x0	OVERRIDE_FOR_3_3V: Maximum override for 3.3V VDD1

32.9.3.11 SDMMCAA_MAX_CURRENT_OVERRIDE_HI_0

Offset: 0x128 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	OVERRIDE_FOR_1_8V_VDD2: Maximum override for 1.8V VDD2

32.9.3.12 SDMMCAA_VENDOR_IO_TRIM_CNTRL_0

Vendor I/O trimmer control register

Used to configure I/O trimmer

Table 208: Truth Table

Input Pins			Output	Comments
E_DPD	SEL_VREG	SEL_VREF	CLKOUT	
1	x[1]	x	0	The cell is in deep power down mode.
0	1	x	based on ip_clk_select selected clock input	Trimmer is powered by VAUXC.
0	0	x	based on ip_clk_select selected clock input	Trimmer is powered by regulated voltage.

1. 'x' indicates don't care.

* 'x' indicates don't care

Offset: 0x1ac I Read/Write: R/W I Reset: 0x00000015 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx010101)

Bit	Reset	Description
5	ENABLE	TRIM_PWRSAVE: Enables power saving mode by clock gating the unused taps in delay chain Active low signal, 0 - power saving mode enabled - clock gating is enabled for unused trimmer taps - may affect tap delay 1 - no power saving - all the trimmer taps are not clock gated 0 = ENABLE 1 = DISABLE
4:3	VREF_925_MV	SEL_VREF_LEVEL: Selects Vref voltage level 0 = VREF_875_MV 1 = VREF_900_MV 2 = VREF_925_MV 3 = VREF_950_MV
2	0x1	SEL_VREG: Select voltage supply for delay chain present in both Trimmer and DLLPROD value: 0x0 ***Software should set this to 0x0 before accessing SD/eMMC. This setting makes IB trimmer delay independent of VDD_CORE*** 0 - selects regulated reference voltage for trimmer and DLL supply - default (recommended option for tunable SD/eMMC modes) 1 - selects VAUXC for trimmer and DLL supply and shut down BG+REG circuit (can be used in non-tunable modes for power saving) Power up time for BG+REG is ~3us(worst case). Power down time for BG+REG is ~1us(worst case). If software wants to turn on/off BG+REG when SDMMC is idle, it has to take hit of 3us power on time. When SEL_VREG is toggled, both DLL and rx clock trimmer output could glitch irrespective of input clock state which could cause corresponding rx CMD and DATA FIFOs to go into bad state. Software should issue SW_RESET_DAT and SW_RESET_CMD to reset host FIFOs after BG <-> VAUXC switching. This would ensure error free data transfers from there on. Powering down BG would need 3 us turn ON time which may cause IOPS reduction, if software shuts down BG after every transfer and enables it on seeing new transfer request; it may not be possible to do dynamic shut down of BG without stalling new requests.
1	0x0	SEL_VREF: Select reference voltage for voltage regulator 0 - selects Bandgap Voltage Reference (recommended option for SD/eMMC tunable modes) 1 - selects resistor divider voltage reference and power down bandgap Switching time between the supplies is 1us. When switching from one supply to other supply, we need to wait for at least 1us before doing any data transfers.
0	0x1	PD_BGREG: Not used - Dummy control Power down Band Gap voltage reference, voltage regulator and resistor chain voltage ref (BG+REG) present in custom I/O trimmer used for SD/eMMC bus tuning. Active High signal, PD_BGREG=1 => Power down BG+REG; PD_BGREG=0 => power up Power down and up time for BG+REG is 1us. If software wants to turn on/off BG+REG when SDMMC is idle, it has to follow 1us power on/off time. Back-up option for powering down BG. Software should clear this bit when it wants to turn ON BG by setting SEL_VREG=0. The original idea to have PD_BGREG pin is to provide power saving feature when the eMMC/SDMMC is in IDLE state, but not in DPD mode. This pin function is actually merged into SEL_VREG function -BG+REG circuit is shut-down when SEL_VREG=1 (trimmer powered by VAUXC)

32.9.3.13 SDMMCAA_VENDOR_DLLCAL_CFG_0

Vendor DLL calibration configuration register

This register is used to setup the DLL Calibration settings. Software has to run DLL calibration first before initiating data transfers in HS400 or HS667 modes and re-calibration is required whenever there is a change in eMMC device clock frequency in HS400/HS667 modes.

Offset: 0x1b0 I Read/Write: R/W I Reset: 0x16083504 (0b0x01011000001000001101010000010 0)

Bit	Reset	Description
31	0x0	CALIBRATE: CALIBRATE is used to start a DLL calibration process. Software should set this bit to trigger calibration and should not clear this bit. This bit will be cleared once the calibration process is completed.
29:25	0xb	END_COUNT: END_COUNT determines the end condition of the calibration. If USE_STATIC_CYCLES is not set, a usec timer is loaded with (2^END_COUNT) and calibration will be stopped once the timer expires. Recommended timer value is 1msec. If USE_STATIC_CYCLES is set and if the trimmer tap value being calibrated has not changed for this number of loops, the calibration ends. The number iterations is (2 ^ END_COUNT), if END_COUNT != 0
24	0x0	USE_STATIC_CYCLES: When set, calibration will be stopped when num static cycles reaches END_COUNT else usec timer is used. Recommended option is use usec timer.
23	0x0	IGNORE_START_TRIM: IGNORE_START is used to ignore the START_TRM value. Instead the calibration starts at the current programmed DDLLCAL value. This is intended to be used if we ever support periodic DDLL calibration.
22:16	0x8	START_TRIM: START_TRM specifies the starting trimmer value to calibrate the with.
15:12	0x3	FILTER_BITS: FILTER_BITS is the LSB of the counter to use for updating the trimmer value. The new trimmer value is trim_old + (count >> FILTER_BITS) - (trim_old >> FILTER_BITS) FILTER_BITS chosen < bits used for SAMPLE_COUNT
11:8	0x5	SAMPLE_COUNT: SAMPLE_COUNT is the number of times the phase detector is sampled before going onto the next set of trimmer values. The number of samples is (2 ^ SAMPLE_COUNT)

Bit	Reset	Description
7:0	0x4	SAMPLE_DELAY: SAMPLE_DELAY is the number of SDMMC host clks from changing the trimmer value to when the phase detector is sampled.

32.9.3.14 SDMMCAA_VENDOR_DLL_CTRL0_0

Vendor DLL control register0

This register has software overrides for controlling both master and slave DLL inputs

Offset: 0x1b4 I Read/Write: R/W I Reset: 0x60000000 (0b01100000000000000000000000000000)

Bit	Reset	Description
31	0x0	MST_DLL_CLK_EN_OVERRIDE: Master DLL CLK_IN enable override 0 = NORMAL: 0 -> Mst DLL clk_in is gated when DLL calibration is not running 1 = OVERRIDE: 1 -> Mst DLL clk_in is not gated irrespective of DLL calibration status
30	0x1	TX_SLV_DLL_PWRSAVE:PWRSAVE is active low signal 1: no clock gating enabled for TX slave DLL 0: clock gating is enabled for unused delay taps in TX slv DLL to save power. This control does not affect the logic output of slave DLL; controls slv dll enable in iobrick
29	0x1	RX_SLV_DLL_PWRSAVE:PWRSAVE is active low signal 1: no clock gating is enabled for RX slave DLL0: clock gating is enabled for unused delay taps in RX slv DLL to save power. This control does not affect the logic output of slave DLL; controls slv dll enable in iobrick
28:22	0x0	TX_SLV_DLL_DLY_CODE:7-bit delay code (128 taps) to be applied to TX slave DLL when DLLCAL_BYPASS is enabled
21:15	0x0	RX_SLV_DLL_DLY_CODE:7-bit delay code (128 taps) to be applied to RX slave DLL when DLLCAL_BYPASS is enabled
14	DISABLED	DLLCAL_BYPASS:DLL calibration bypass enable 0 = DISABLED: 1: Programmed delay code will be applied to both master and slave DLLs; 1 = ENABLED: 0: Delay code determined by DLL calibration is applied to master and slave DLLS
13:7	0x0	TX_DLY_CODE_OFFSET: two-s complement offset will be added to delay code generated by calibration controller and sent to TX slv DLL
6:0	0x0	RX_DLY_CODE_OFFSET: two-s complement offset will be added to delay code generated by calibration controller and sent to RX slv DLL

32.9.3.15 SDMMCAA_VENDOR_DLL_CTRL1_0

Vendor DLL control register1

This register has software overrides for controlling both master and slave DLL inputs.

Offset: 0x1b8 I Read/Write: R/W I Reset: 0x20208780 (0bx0100000x01000001000011110000000)

Bit	Reset	Description
30:24	0x20	REQD_DELAY_STEPS_TX: Delay required in steps of 1/64 UI - 0 to 63 steps MST DLL is half cycle locked - UI=0.5 cycle=64steps. Default value is quarter cycle. Calibrated code covers one UI of MST DLL. This reg field value is used to scale the delay code before sending it to TX slv DLL.
22:16	0x20	REQD_DELAY_STEPS_RX: Delay required in steps of 1/64 UI - 0 to 63 steps MST DLL is half cycle locked - UI=0.5 cycle=64 steps. Default value is quarter cycle. Calibrated code covers one UI of MST DLL. This reg field value is used to scale the delay code before sending it to RX slv DLL.
15:11	0x10	MST_DLL_RESET_TIME: Master DLL reset duration - 16 cycles
10:7	0xf	SLV_DLL_SETTLE_TIME: Slave DLL requires 4 cycles settle time when dly code is changed for providing stable output
6:0	0x0	MST_DLL_DLY_CODE:7-bit delay code (128 taps) to be applied to master DLL when DLLCAL_BYPASS is enabled

32.9.3.16 SDMMCAA_VENDOR_DLLCAL_CFG_STA_0

Vendor DLL calibration configuration register

This register is used to setup the DLL Calibration settings.

Software has to run DLL calibration first before initiating data transfers in HS400 or HS667 modes and re-calibration is required whenever there is a change in eMMC device clock frequency in HS400/HS667 modes. DLL calibration process is an open loop process in which DLL calibration controller can’t decide the final tap value on its own by looking at PD output from DLL.

Summary of DLL calibration process implemented:

1. Software programs SAMPLE_COUNT (number of samples to be taken per every tap value) and STOP_TIMER_VALUE in microseconds.
2. These two values decide the stop condition of calibration.
3. DLL calibration is triggered on writing '1' into CALIBRATE register field by software. The controller sets CAL_ACTIVE bit to indicate calibration running status.
4. DLL controller starts calibration by applying a tap value (may use START_TAP programmed) to MST DLL and loads usec timer with STOP_TIMER_VALUE.
    a. After SAMPLE_DELAY (4 cycles of DLL in clock), it starts taking SAMPLE_COUNT number of PD samples. A phase counter is updated based on PD value per every cycle.
    b. Based on phase counter value, 'Filter' decides whether to increment or decrement current tap value to get next tap value to be applied in next iteration.
    c. FILTER_BITS decides the accuracy of the decision made. FILTER_BITS < bits used for SAMPLE_COUNT.
5. Repeat step 3 till STOP_TIMER expires.
6. The controller stops calibration (updating tap value) and latches the tap value used in last calibration iteration. CALIBRATE bit will be cleared at this step.
7. The controller adds offsets programmed to calibrated tap value and applies the final codes to SLAVE DLLs.
8. The controller waits for slave DLL settle time and clears CAL_ACTIVE register bit.
9. Software polls for CAL_ACTIVE=0 and can start data transfers.

DLL lock time is represented as:

Each tap test time = (SAMPLE_DELAY + SAMPLE_COUNT) * DLL_CAL_CLK_PERIOD

Worst case DLL locking time = STOP_TIMER_VALUE = (NUM_CAL_ITERATIONS * (SAMPLE_DELAY + SAMPLE_COUNT) * DLL_CAL_CLK_PERIOD)

Default values chosen:

Worst case DLL locking time=1 ms = 1000 us = STOP_TIMER_VALUE.

SAMPLE_COUNT=32 samples per one tap;

SAMPLE_DELAY=4 cycles

Tap value is decided by PD and filter logic.

Notes:

- *Software needs to calibrate DLL before using eMMC5.0 HS400/HS667 mode (before tuning process). Re-calibration is required whenever eMMC I/O clock frequency is changed. This can be done as a part of standard clock frequency change sequence.*
- *DLL calibration would work at when I/O CLK is running at 200 MHz and above.*
- *For I/O CLK frequency is below 200 MHz, software should calculate delay code as per below equation:*
- *Delay Code = (Delay Code at 200 MHz) * 200 MHz / Target Frequency*
- *Under 85 MHz, DLL delay code may reach its maximum code (128 taps) for FF chips. In this case, software should use 128 taps.*
- *Software can run DLL calibration at 200 MHz once at boot time, save those values, scale and use them whenever frequency is changed. This is needed if software wants to do DFS.*

Offset: 0x1bc I Read/Write: R/W I Reset: 0xX0XXXXXX (0bxx010000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RO	X	DLL_CAL_ACTIVE: Calibration process status software has to wait for DONE once calibrate bit is cleared and before starting data transfers on eMMC interface. 0 = DONE: 0: DLL calibration is completed - calibration engine is idle and slave DLLs are ready for normal operation 1 = RUNNING: 1: DLL calibration is still running
30	RO	X	DLL_PD: Master DLL Phase detector output
29	RW	0x0	MST_DLL_RST_OVERRIDE_EN: When set, master dll reset is controlled by programming MST_DLL_RST_ Else reset is generated by DLL controller
28	RW	0x1	MST_DLL_RST_: Master DLL reset - active low signal - used when DLL_RST_OVERRIDE_EN is set minimum reset duration is 16 sdhost clk cycles
27	RW	0x0	SLV_DLL_CLK_OUT_DIS_OVERRIDE_EN: When set, slave dll clk_out_dis is controlled by programming SLV_DLL_CLK_OUT_DIS Else clk_out_dis is controlled by DLL controller
26	RW	0x0	SLV_DLL_CLK_OUT_DIS: Slave DLL clock out disable - used when SLV_DLL_CLK_OUT_DIS_OVERRIDE_EN is set 0: Delay line clock normal output1: DelayLIne clock output gated and grounded to 0.should be set to 1 when software tries to update slv dll delay code should be cleared after slave dll settle time - 3cycles
25	RW	0x0	MST_DLL_PWRDN_OVERRIDE_EN: When set, master dll can be kept in power down mode by programming MST_DLL_PWRDN field. Else pwrdn is controlled by DLL controller
24	RW	0x0	MST_DLL_PWRDN: Master DLL power down enable - active high control - used when MST_DLL_PWRDN_OVERRIDE_EN is set 0: power up; 1 - power down. Software can keep MST DLL in power down mode once calibration is done.MST DLL should be powered up before triggering calibration. Controls dll_en in iobrick
22:16	RO	X	TX_DLY_CODE_ADJ: delay code sent to TX slave DLL after applying offset; Valid only when DLL auto-calibration is used. (calib_dly_code * reqd_dly_steps/64) + offset
14:8	RO	X	RX_DLY_CODE_ADJ: delay code sent to RX slave DLL after applying offset; Valid only when DLL auto-calibration is used. (calib_dly_code * reqd_dly_steps/64) + offset
6:0	RO	X	DLY_CODE: Holds delay code determined by calibration process which is sent to MST DLL during calibration; Valid only when DLL_CAL_ACTIVE is set

32.9.3.17 SDMMCAA_VENDOR_TUNING_CNTRL0_0

Vendor Tuning Control0 Register

Offset: 0x1c0 I Read/Write: R/W I Reset: 0x74020090 (0bx11101000000001x0000000010010000) I Default: 0x00000040

Bit	Reset	SW Default	Description
30	0x1	_NONE_	RD_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on read data CRC error
29	0x1	_NONE_	WR_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on write CRC error
28	0x1	_NONE_	CMD_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on cmd CRC error
27	0x0	_NONE_	RETUNING_REQ_EN_ON_CRC_ERR_DETECTION: Re-tuning request is generated when set to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
26	0x1	_NONE_	TUNING_ERR_EN_ON_CRC_ERR_DETECTION: Tuning error is generated to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
25:18	0x0	_NONE_	START_TAP_VAL: start tap value to be used by tuning; start_tap should be multiple of step_size chosen and its valid range is 0-255; Not valid when TAP_VAL_UPDATED_BY_HW is set to zero. Tuning algorithm uses this as the start tap value for scanning through trimmer taps.
17	0x1	_NONE_	TAP_VAL_UPDATED_BY_HW: This bit is functional only in tunable modes (SDR50, SDR104 and HS200) software can choose to update the tap value by itself by clearing this bit; Preferred value is 1 - tap value is updated by hardware. If this bit is cleared, hardware does not update tap_val per every tuning iteration. Software can update it as desired. Tuning pattern match is indicated by sampling_clock_select per every tuning iteration. Software has to maintain the status of each tuning iteration and determine the best PASS window to fix the final sampling point. Software can program NUM_TUNING_ITERATIONS as desired. Once the number of tuning commands issued reaches number of tuning iterations programmed, execute_tuning bit will be cleared to indicate the completion of tuning procedure. Note that using this option violates Host specification but provided for legacy reasons. It helps us in using legacy software tuning solution in case hardware solution does not work.

Bit	Reset	SW Default	Description
15:13	TRIES_40	_NONE_	NUM_TUNING_ITERATIONS: The number of tuning iterations to be used by tuning circuit; Preferred value is 40 iterations as per Host specification. Using other values violates the specification but provided for trimmer characterization purpose. 0 = TRIES_40 1 = TRIES_64 2 = TRIES_128 3 = TRIES_192 4 = TRIES_256
12:6	0x2	0x1	MUL_M: implements a multiplier - M+1 Final tap value is derived from best passing window and calculated as follows. Final tap value = first_pass + ((last_pass - first_pass)*Q); where Q = percentage of pass window;default-75% Q = M+1/(2^N); N:1...7 M:should be in range [0:2^N-1];
5:3	0x2	_NONE_	DIV_N: implements a divider - 2^N; maximum div is 2^7 =>128
2:0	0x0	_NONE_	TUNING_WORD_SEL: Selects desired word from 256-bit tuning status bitmap status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.3.18 SDMMCAA_VENDOR_TUNING_CNTRL1_0

Vendor Tuning Control1 register

Different step size is required in SDR50 mode to cover two UI (100 MHz => 2*10ns)

With 70ps/tap trimmer resolution, we can cover almost 2UI using step_size=8 in SDR50 and step_size=4 in SDR104.

Tuning will be done in HS200 - SDR mode only.

HS200 - tuning at 200 MHz - UHS_SEL should be SDR104 for executing tuning

Before initiating data transfers in HS400 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 200 MHz

For overclocked modes: Before initiating data transfers in HS667 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 333 MHz

---

Note: *eMMC bus tuning will always be run in HS200 mode only.*

Use step_size = 1 to scan through all taps.

Offset: 0x1c4 I Read/Write: R/W I Reset: 0x00000023 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010x011) I Default: 0x00000000

Bit	Reset	SW Default	Description
6:4	0x2	0x0	STEP_SIZE_SDR104_HS200: tap_val is incremented by step_size for every tuning iteration - used in SDR104/ HS200/HS400 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD For overclocking modes, set this field to 0x1
2:0	0x3	0x0	STEP_SIZE_SDR50: tap_val is incremented by step_size for every tuning iteration - used in SDR50 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD

32.9.3.19 SDMMCAA_VENDOR_TUNING_STATUS0_0

Vendor Tuning Status0 register

Offset: 0x1c8 I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	STATUS_WORD: Each bit indicates the status of each tuning iteration, when tap value is updated by hardware (TAP_VAL_UPDATED_BY_HW=1); 0-Tuning pattern not matched 1-tuning pattern matched We have a total of 256 tap values. Software can issue a maximum of 256 tuning commands for debug. Software needs to read this register eight times to get status of all 256 iterations by changing tuning_word_sel status[255:0] is left shifted and loaded into this register every time when tuning_word_sel is changed status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.3.20 SDMMCAA_VENDOR_TUNING_STATUS1_0

Vendor Tuning Status1 register

Offset: 0x1cc I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	PASS_WINDOW_END_BEFORE_FINE_TUNING: End tap value of best PASS window found by scan FSM
23:16	X	PASS_WINDOW_START_BEFORE_FINE_TUNING: Start tap value of best PASS window found by scan FSM
15:8	X	PASS_WINDOW_END_AFTER_FINE_TUNING: End tap value of best PASS window after fine tuning
7:0	X	PASS_WINDOW_START_AFTER_FINE_TUNING: Start tap value of best PASS window after fine tuning

32.9.3.21 SDMMCAA_VENDOR_CLK_GATE_HYSTERESIS_COUNT_0

Vendor Clk gating Hysteresis Counter initial value

Offset: 0x1d0 I Read/Write: R/W I Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx001111)

Bit	Reset	Description
5:0	0xf	CLK_COUNT: Before gating second level clocks controller will wait for these many cycles. we can recover if any idle windows are missed in clken equation

32.9.3.22 SDMMCAA_VENDOR_PRESET_VAL0_0

Vendor Preset Value Registers

SD host specification defines one preset value register for each bus speed mode which should be set by host by some unique method.

Preset values vary based on the base frequency used which is in software (Tegra system driver) control.

Tegra System driver supposed to set BASE_CLK_FREQ in VENDOR_CLOCK_CNTRL register before handling over the control to SD host standard driver.

In the similar way, system driver should set below vendor preset values based on the base clock frequency and the desired card clock frequency in each bus speed mode

This should be done after every time SDMMC is reset and after every soft reset.

This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.

Default values are set assuming base clock frequency=208MHz.

Offset: 0x1d4 I Read/Write: R/W I Reset: 0x00201000 (0bxx00000001000000001000000000000)

Bit	Reset	Description
29:20	0x2	SDCLK_FREQ_SEL_HIGH_SPEED: System software programs 10-bit divider value to generate SDCLK in default speed mode (<50 MHz, 3.3V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
19:10	0x4	SDCLK_FREQ_SEL_DEFAULT: System software programs 10-bit divider value to generate SDCLK in default speed mode (<25 MHz,3.3V signaling). This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x4 assuming 208 MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_INIT: System software programs 10-bit divider value to generate desired SDCLK frequency during initialization. This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_LOW register field. For example, if 400 KHz SDCLK is desired at base clk freq=48 MHz, this register should be programmed with 0x3C

32.9.3.23 SDMMCAA_VENDOR_PRESET_VAL1_0

Offset: 0x1d8 I Read/Write: R/W I Reset: 0x00100804 (0bxx00000000100000000100000000100)

Bit	Reset	Description
29:20	0x1	SDCLK_FREQ_SEL_SDR50: System software programs 10-bit divider value to generate SDCLK in SDR50 mode (<100MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x1 (gives 2N divider) assuming 208 MHz base clock
19:10	0x2	SDCLK_FREQ_SEL_SDR25: System software programs 10-bit divider value to generate SDCLK in SDR25 mode (<50 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
9:0	0x4	SDCLK_FREQ_SEL_SDR12: System software programs 10-bit divider value to generate SDCLK in SDR12 mode (<25 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_HIGH register field

32.9.3.24 SDMMCAA_VENDOR_PRESET_VAL2_0

Offset: 0x1dc I Read/Write: R/W I Reset: 0x00000800 (0bxxxxxxxxxxxxx00000000100000000000)

Bit	Reset	Description
19:10	0x2	SDCLK_FREQ_SEL_DDR50: System software programs 10-bit divider value to generate SDCLK in DDR50 mode (<50 MHz,1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x2 assuming 208 MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_SDR104: System software programs 10-bit divider value to generate SDCLK in SDR104 mode (<208 MHz, 1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_LOW register field

32.9.3.25 SDMMCAA_SDMEMCOMPPADCTRL_0

SDMEMCOMP Pad control register

This register is used to control COMP pad inputs.

Offset: 0x1e0 I Read/Write: R/W I Reset: 0x88000001 (0b10x01xxxxxxxxxxxxxxxx0xxx0000001) I Default: 0x00000000

Bit	Reset	SW Default	Description
31	0x1	0x0	PAD_E_INPUT_OR_E_PWRD: used to control E_INPUT (for SDMMC1/3) and E_PWRD (for SDMMC2/4) input of pu/pd comp pad should be set before starting auto-cal and cleared once auto-calibration is done (for power saving) NOTE: E_PWRD = !PAD_E_INPUT_OR_E_PWRD and E_INPUT = PAD_E_INPUT_OR_E_PWRD
30	0x0	_NONE_	COMP_PAD_REG_ON: If set, turns ON regulator in comp pad. Software should not set this bit.
28:27	0x1	_NONE_	COMP_PAD_DRV_TYPE: used to control drv_type input of BDSDMEMLVCOMP_C pad
10	0x0	_NONE_	COMP_PAD_E_TEST_OUT: used to control e_test_out input of COMP pad
6:4	0x0	_NONE_	COMP_PAD_TEST_SEL: used to control test_sel input of COMP pad
3:0	0x1	_NONE_	SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL:used to control vref_sel input of COMP pad. Software should set this to 0x7.

32.9.3.26 SDMMCAA_AUTO_CAL_CONFIG_0

SDMEMCOMP pad auto-calibration settings

AUTO_CAL_SLW_OVERRIDE

0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output

DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3]

1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs

AUTO_CAL_OVERRIDE

0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting

1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly

Offset: 0x1e4 I Read/Write: R/W I Reset: 0x00010000 (0b0000xxxxxxxxx001x0000000x0000000) I Default: 0x20000000

Bit	Reset	SW Default	Description
31	0x0	_NONE_	AUTO_CAL_START: Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	0x0	_NONE_	AUTO_CAL_OVERRIDE: AUTOCAL override. 0 = NORMAL: 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 = OVERRIDE: 1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly
29	DISABLED	0x1	AUTO_CAL_ENABLE: AUTOCAL enable. 0 = DISABLED: 0 (disabled): use sdmmc2tmc_cfg* register settings for pullup/dn 1 = ENABLED: 1 (normal operation): use SDMMC generated pullup/dn (override or AUTOCAL)
28	0x0	_NONE_	AUTO_CAL_SLW_OVERRIDE: AUTOCAL slew rate override 0 = NORMAL: 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/ UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3] 1 = OVERRIDE: 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs
18:16	0x1	_NONE_	AUTO_CAL_STEP: calibration step interval (in microseconds)
14:8	0x0	_NONE_	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
6:0	0x0	_NONE_	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

32.9.3.27 SDMMCAA_AUTO_CAL_INTERVAL_0

SDMEMCOMP pad calibration interval - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1e8 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	AUTO_CAL_INTERVAL: 0: do calibration once Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.This feature is not supported from Tegra X1 due to movement of level shifters to COMP pad.Software should trigger calibration at regular intervals, if needed.

32.9.3.28 SDMMCAA_AUTO_CAL_STATUS_0

SDMEMCOMP pad calibration status - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1ec I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (AUTO_CAL_ACTIVE == 0)
30:24	X	AUTO_CAL_PULLDOWN_ADJ: Pulldown code sent to pads
22:16	X	AUTO_CAL_PULLUP_ADJ: Pullup code sent to pads
14:8	X	AUTO_CAL_PULLDOWN: Pulldown code generated by auto-calibration
6:0	X	AUTO_CAL_PULLUP: Pullup code generated by auto-calibration

32.9.3.29 SDMMCAA_IO_SPARE_0

SPARE bits. These SPARE_OUT bits go to pipe -> pad and then come back as SPARE_IN

SPARE_OUT[3]: IO_SPARE[19] - used as MUX select which selects between one cycle delay and two cycle delay versions of cmd_oen to mask wdata of IB capture flop.

0x0: selects two cycle delayed version

0x1: selects one cycle delayed version - recommended

SPARE_OUT[2]: IO_SPARE[18] - used as active low enable for gating both CMD_IN and DAT_IN async FIFOs wdata when we are driving CMD/DAT lines.

0x0: write 1 when OEN is active and Zi value when OEN is not active into FIFO (default)

0x1: write Zi value into FIFO irrespective of OEN state.

Offset: 0x1f0 I Read/Write: R/W I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx) I Default: 0x00080008

Bit	R/W	Reset	SW Default	Description
31:16	RW	X	0x8	SPARE_OUT:PROD value: Software should set SPARE_OUT[3]: IO_SPARE[19] to 1 before accessing SD/eMMC.
15:0	RO	X	0x8	SPARE_IN

32.9.3.30 SDMMCAA_SDMMCAA_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register.

Note: *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility*

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and mc side of the mccif. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clk is enabled.

With wclk/rclk override mode = ON, the clock is always on inside mccif and PC.

A '1' written to the cclk override field keeps client's clk always on inside mccif.

Offset: 0x1f4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	0x0	SDMMCA_RCLK_OVR_MODE
19	0x0	SDMMCA_WCLK_OVR_MODE
18	0x0	SDMMCA_CCLK_OVERRIDE
17	0x0	SDMMCA_RCLK_OVERRIDE
16	0x0	SDMMCA_WCLK_OVERRIDE

32.9.3.31 SDMMCAA_TIMEOUT_WCOAL_SDMMCAA_0

Write Coalescing Time-Out Register

Note: *Write coalescing is no longer supported by the MCCIF clients.*

Registers are retained for software compatibility but are not used by the hardware.

Offset: 0x1f8 I Read/Write: R/W I Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	SDMMCWA_WCOAL_TMVAL

32.9.4 SDMMC3 Vendor Registers

32.9.4.1 SDMMC_VENDOR_CLOCK_CNTRL_0

The following registers are Vendor Specific Registers and are mapped to Vendor Specific Address Space (0x100 - 0x1FF)

Vendor Clock Control Register

Offset: 0x100 I Read/Write: R/W I Reset: 0x0000d02d (0bxx00000000000000011010000x0101101)

Bit	Reset	Description
29	0x0	DIFF_CLK_SEL: If set, selects differential CLK and DQS; Used by eMMC IOBRICK only. Software should set this appropriately based on eMMC part used. E_INPUT_* (CMD/DAT/CLK) of emmc IOBRICK should be set to 0 when differential signaling is used. default is '0' - selects single ended signaling for clk/dqs
28:24	0x0	TRIM_VAL: Tap value for output data path trimmer This determines the trimmer value needed to drive the output data correctly. The tap for outbound trimmer is single MUX. The trim settings required are within very small (0-3) with absolute delay requirement of ~400ps. Minimal change with PVT variations. Following values are recommended. These are common for all speed modes of SD/eMMC. These are the initial values to start with. SDMMC1 - 2 SDMMC2 - 8 SDMMC3 - 3 SDMMC4 - 8
23:16	0x0	TAP_VAL: Tap value for input data path trimmer This determines the tap value needed to sample the input data correctly. Delay per each tap can range from 70ps (hv_ff) to 505ps (lv_ss). Following values are recommended. These are the initial values to start with. 1> SDR12/SDR25/High Speed DDRmodes: If I/O pad trimmer is used (default): SDMMC1 - 4 SDMMC2 - 0 SDMMC3 - 3 SDMMC4 - 0 If core legacy trimmer is used: SDMMC1 - 3 SDMMC2 - 3 SDMMC3 - 3 SDMMC4 - 3 2> SDR50 and SDR104/HS200/HS400/HS667 should use tuning procedure to determine the tap value
15:8	0xd0	BASE_CLK_FREQ: System software programs the base SD/MMC clock frequency into this register before loading the SD/MMC driver.This should not be touched when software wants to use SD card in uhs-II mode. This value is readable in the standard, but not writeable, SDMMC_CAPABILITIES_0_BASE_CLOCK_FREQUENCY register field. Software should program the actual clock frequency programmed in CAR registers CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC*_0 for each SDMMC controller. This should be done after every time SDMMC is reset and after every soft reset. This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.
6	0x0	LEGACY_CLKEN_OVERRIDE: Override for sdmmc_legacy_g_clk clken; Set this to 0 when in UHS-II mode to save power 0 = NORMAL: 0 -> sdmmc_legacy_g_clk is gated 1 = OVERRIDE: 1 -> sdmmc_legacy_g_clk is not gated
5	0x1	SDR50_TUNING_OVERRIDE: override the SDR50_TUNING capabilities bit. Software should only set this bit if it is required to use Tuning for SDR50. (only supported for SDMMC1 and SDMMC3) 0 = NORMAL: 0 -> No Tuning support advertised for SDR50 mode. 1 = OVERRIDE : 1 -> Tuning support is enabled for SDR50 mode.
4	0x0	UHS2_CAPABILITY_OVERRIDE: override the UHS-II capabilities bit. 0 = NORMAL: 0 -> UHS-II support is unadvertised 1 = OVERRIDE: 1 -> UHS-II support is advertised
3	0x1	PADPIPE_CLKEN_OVERRIDE: Override for padmacro and pipemacro clken. 0 = NORMAL: 0 -> CLKEN is de-asserted when internal CLKEN is de-asserted. 1 = OVERRIDE : 1 -> CLKEN is kept asserted even when internal CLKEN is de-asserted. Software should always program this to 0x1 (OVERRIDE).

Bit	Reset	Description
2	0x1	SPI_MODE_CLKEN_OVERRIDE: Override for CLKEN during SPI_MODE during resettle. 0 = NORMAL: 0 -> CLKEN is de-asserted while doing sw_reset. 1 = OVERRIDE: 1 -> CLKEN is kept asserted while doing sw_reset. Software should always program this to 0 (NORMAL).
1	0x0	INPUT_IO_CLK: Feedback clock is selected by default. Software should not change this. Disabling Feedback clock will select Internal Clock that requires different TAP Value Programming. 0 = FEEDBACK 1 = INTERNAL
0	0x1	SDMMC_CLK: This is set when sdmmc_clk is supplied by the CAR module.Prior to sdmmc_clk switch OFF.This bit should be written '0'. By writing zero,the asynchronous card interrupt is routed to the Interrupt controller. 0 = DISABLE 1 = ENABLE

32.9.4.2 SDMMC_VENDOR_SYS_SW_CNTRL_0

Vendor System Software Control Register

Offset: 0x104 I Read/Write: R/W I Reset: 0x38600002 (0b00111000011xxxxxx0000000x0000010)

Bit	Reset	Description
31	0x0	ENHANCED_STROBE_MODE: Enables enhanced strobe mode in HS400/HS667 mode for eMMC5.x devices 0 - cmd_in(Resp) is sampled by loopback clock which requires tuning 1 - cmd_in(Resp) is sampled by DQS_in which requires no tuning software has to set this bit appropriately based on device capability since this is an optional feature for eMMC5.x devices. If device supports this feature, software should set this bit to avoid tuning.
30	0x0	USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT: When set, uses TMCLK data timeout counter for generating wr_crc_status data-timeout When cleared, uses sdmmc_clk for maintaining wr_crc_status data timeout counter
29	0x1	USE_NCRC_FOR_WR_CRC_STATUS_TIMEOUT_VAL: This field is valid only when USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT is set to 0. When cleared, uses data timeout value as wr CRC status timeout value (specification defined one) When set, uses Ncrc cycles as timeout value
28	0x1	USE_TMCLK_FOR_DATA_TIMEOUT: When set, uses TMCLK data timeout counter for generating legacy data timeout error (except wr_crc_status timeout) When cleared, uses sdmmc_clk for maintaining data timeout counter
27:24	0x8	DEVICE_BUSY_WAIT_CYCLES: In HS400/667 mode, busy should be sampled using clock instead DQS.So, Dat0 from card is passed to two different async FIFOs (one is running in sdmmc_clk and other is in sdmmc_dqs).CRC status is taken from dat0 sampled in DQS and busy should be checked after CRC status end bit reception using dat0 sampled in clk.Both clk and DQS dat0 paths don't have same prop delay due to sync-ers used in async FIFOs and also due to phase difference between clk and dqs.Due to this delay difference between the paths, we may sample wrong busy status if we sample busy immediately after receiving END bit of CRC status. To avoid incorrect sampling of busy, a wait state before busy polling is added. This register field is used to load wait_cycles counter. Note that this counter is used only in HS400/667 mode.
23	0x0	ALLOW_CARD_CLK_STALLS_IN_WR: When set, allows card clock stopping during transfer of data within a block in HIGH SPEED DDR/HS400 writes.
22	0x1	EMMC_IOBRICK_CLK_DATA: Used to drive AP_CLK and AN_CLK input of iobrick.0x1 - clk_out will be same as iobrick_clk_in0x0 - clk_out will be inverted iobrick_clk_in
21	0x1	QUALIFY_WITH_RD_DATA_VLD: We have async FIFOs in both cmd_in and dat_in paths in padmacro which are used in tunable modes.When this bit set, rdata from FIFO is treated as valid data only when rd_req is high. This is needed to handle bubbles on 'rd_req' when MTBF is high.
14	0x0	SD_BUS_POWER_ON_OFF_INT_STATUS: SD_BUS_POWER was changed. System software can use this interrupt to implement power switch.
13	0x0	VOLT_SWITCH_INT_STATUS: VOLT_18_EN was changed. System software can use this interrupt to implement a UHS-I voltage switch procedure for a standard SD Host driver 0 = NO_INT 1 = GEN_INT
12	0x0	TUNING_SYS_INT_STATUS: CMD19 was issued while EXECUTE_TUNING was set. System software can use this interrupt to implement a UHS-I tuning procedure for a standard SD Host driver. 0 = NO_INT 1 = GEN_INT
11:8	0x0	TUNING_ASYNC_FIFO_ADDNL_DELAY: In tunable modes, data/cmd IB path uses async FIFO which contributes additional delay to cmd/resp and wdata/CRC token round trip delay.Controller waits for this round trip delay before waiting for Ncr or Ncrc. This is required not to latch previous data/cmd driven by host itself as start bits of resp or CRC.Round trip delay is calculated for async FIFO with sync-2d.This register field holds the additional delay in cycles which should be added to round trip delay. This additional delay is caused by increasing sync-er depth (more than 2) or addition of output flop stage in async FIFO. Default value is zero. NOTE: Software should not update this field unless a new PROD setting is given.

Bit	Reset	Description
6	0x0	SD_BUS_POWER_ON_OFF_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when SD_BUS_POWER is changed. 0 = DISABLE 1 = ENABLE
5	0x0	VOLT_SWITCH_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when VOLT_18_EN is changed. 0 = DISABLE 1 = ENABLE
4	0x0	TUNING_SYS_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when CMD19 is issued while EXECUTE_TUNING is set. 0 = DISABLE 1 = ENABLE
3	0x0	ASSERT_BUFF_RD_RDY_INT: Write a 1 to this field to assert sdmmc_interrupt_status_0_buffer_read_ready. Used by the system software that implements the tuning procedure to signal to the standard SD driver that the tuning process has completed 0 = DISABLE 1 = ENABLE
2	0x0	IO_TRIM_BYPASS: Override bit for selecting between core trimmer (Vcore dependent) and I/O trimmer (custom trimmer) in IB clock path. Default option is I/O trimmer; software should not set this field.
1	0x1	INT_MASK_WHILE_TUNING: As per specification, Host should not generate any interrupts (including cmd_complete and data_xfer_complete) except buffer_read_ready interrupt during tuning sequence is being performed software can override this behavior by clearing this bit - but this leads to a specification violation 0 = DISABLE 1 = ENABLE
0	0x0	SPI_MODE: This is a mirror bit.The SPI mode is set if this bit is set or CMD_XFER_MODE[7] is set Writing 1 will drive the CS Low and writing zero will de-assert the CS Signal 0 = DISABLE: 0 -> SPI mode is disabled. 1 = ENABLE: 1 -> SPI mode is enabled.

32.9.4.3 SDMMC_VENDOR_ERR_INTR_STATUS_0

Legacy Interrupt Status Register

The fields are valid when an error interrupt has occurred.

Offset: 0x108 I Read/Write: RO I Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	SDMMC_LEGACY_CTLR_IDLE: indicates legacy SD interface controller is idle - no active data transfers on legacy SD interface
17	X	READ_DATA_TIMEOUT: valid when a data timeout error occurs
16	X	WRITE_CRC_STATUS_TIMEOUT: valid when a data timeout error occurs
15	X	WRITE_BUSY_TIMEOUT: valid when a data timeout error occurs
14	X	RESP_BUSY_TIMEOUT: valid when a data timeout error occurs
13	X	SPI_WRITE_BUSY_TIMEOUT
12	X	SPI_RX_START_TOKEN_TIMEOUT
8:5	X	SPI_DAT_ERR_TOKEN: Data Error Token, while read from card.
4:0	X	SPI_DAT_RESPONSE: Data Response while write to card 5 = DATA_ACCEPTED 11 = CRC_ERR 13 = WRITE_ERR

32.9.4.4 SDMMC_VENDOR_CAP_OVERRIDES_0

Capabilities override bits

Offset: 0x10c I Read/Write: R/W I Reset: 0x00000007 (0bxxx0xxxxxxxxxxxxx000000xxxx0111)

Bit	Reset	Description
28	0x0	CMD_QUEUING_MODE_EN: useful for eMMC5.x devices which support CMD queuing. Software should enable this when it wants to use cmd queuing of eMMC5.x devices in HS400 mode.
13:8	0x0	DQS_TRIM_VAL:PROD_VAL=0x11 - Tap value for incoming DQS path trimmer - used in HS400/HS667 modes
3	0x0	DRV_LPBK_CLK_ON_CMD_LINE:Loopback trimmed clock will be driven onto cmd line, if this bit set to 1. Should be set to zero during normal data transfers.Useful in debug.
2	0x1	VOLTAGE_3_3_V_SUPPORT_OVERRIDE: Voltage support 3_3_V override
1	0x1	VOLTAGE_3_0_V_SUPPORT_OVERRIDE: Voltage support 3_0_V override
0	0x1	VOLTAGE_1_8_V_SUPPORT_OVERRIDE: Voltage support 1_8_V override

32.9.4.5 SDMMC_VENDOR_BOOT_CNTRL_0

Vendor Boot Control Register

This register is used to configure Boot Mode to support MMC v4.3 cards.

Offset: 0x110 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BOOT_ACK: This bit is used to support Boot Option in MMC 4.3 version cards. If set Boot acknowledgment is given by card else not given by card 0 = DISABLE 1 = ENABLE
0	0x0	BOOT: This bit enables/disable BootOption1.If set BootOption1 is enable, hardware auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

32.9.4.6 SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 0x114 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles. Boot Acknowledgment Timeout error occurs(VENDOR_SPECIFIC_ERR[0])

32.9.4.7 SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 0x118 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles. Then Data Timeout error occurs.

32.9.4.8 SDMMC_VENDOR_DEBOUNCE_COUNT_0

Debounce Counter Value Register

The Debounce Counter runs on 32 kHz clock. Keeping the default value to 100 ms = (100 * 32 cycles/1ms) = 3200 cycles for 100 ms = 0xC80

Offset: 0x11c I Read/Write: R/W I Reset: 0x00000c80 (0bxxxxxxxx00000000000110010000000)

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32KHz clock cycles is programmed to meet Debounce period of the card slot.

32.9.4.9 SDMMC_VENDOR_MISC_CNTRL_0

Miscellaneous Vendor Control Register

SDMMC_SPARE0: Spare register bits with reset value of 0

SDMMC_SPARE0[0]: SW_RESET_CLKEN_OVERRIDE, override the sdmmc_clken when doing SW_RESET if set to 1.

SDMMC_SPARE0[1]: When set, allows SD clock to be stopped in the middle of a read data block while in SDR104/ HS400/HS667 modes(allow_sdr104_intrablock_stalls).

Unsafe for at least some SD/eMMC cards, but may improve SDR104 DMA read performance in some cases.

SDMMC_SPARE0[2]: When set, SDR104 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR104

SDMMC_SPARE0[3]: When set, SDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR50

SDMMC_SPARE0[5]: When 0, masks the pad macro's "high speed" enable to 0, causing the pad macro to always launch data on the falling edge of the clock. This prevents the SD Host driver's setting of SDMMC_POWER_CONTROL_HOST_x_HIGH_SPEED_EN from undesirably affecting the output timing.

SDMMC_SPARE0[7:6]: Number of pipe stages.

SDMMC_SPARE0[8]: When set, DDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_DDR50

SDMMC_SPARE1: Spare register bits with reset value of 1

SDMMC_SPARE1[0]: CMD_CONFLICT_DISABLE, disable command line conflict if set to 1.

SDMMC_SPARE1[1]: Control bit to select between internal and external loopback clock. 1-external 0-internal

Tegra X1 does not support external loopback for any SDMMC controller.

Offset: 0x120 I Read/Write: R/W I Reset: 0xffff0098 (0b11111111111111111000000010011000)

Bit	Reset	Description
31:16	0xffff	SDMMC_SPARE1: Spare register bits with reset value of 1
15:1	0x4c	SDMMC_SPARE0: Spare register bits with reset value of 0x40
0	0x0	ERASE_TIMEOUT_LIMIT: Erase timeout value. 0 = FINITE: Finite, It is limited to the programmed value in the DATA_TIMEOUT_VALUE 1 = INFINITE: Infinite,Controller would be monitoring until the card is busy.

32.9.4.10 SDMMC_MAX_CURRENT_OVERRIDE_0

Offset: 0x124 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	OVERRIDE_FOR_1_8V: Maximum override for 1.8V VDD1
15:8	0x0	OVERRIDE_FOR_3_0V: Maximum override for 3.0V VDD1
7:0	0x0	OVERRIDE_FOR_3_3V: Maximum override for 3.3V VDD1

32.9.4.11 SDMMC_MAX_CURRENT_OVERRIDE_HI_0

Offset: 0x128 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	OVERRIDE_FOR_1_8V_VDD2: Maximum override for 1.8V VDD2

32.9.4.12 SDMMC_VENDOR_IO_TRIM_CNTRL_0

Vendor I/O trimmer control register

Used to configure I/O trimmer

Table 209: Truth Table

Input Pins			Output	Comments
E_DPD	SEL_VREG	SEL_VREF	CLKOUT	
1	x[1]	x	0	The cell is in deep power down mode.
0	1	x	based on ip_clk_select selected clock input	Trimmer is powered by VAUXC.
0	0	x	based on ip_clk_select selected clock input	Trimmer is powered by regulated voltage.

1. 'x' indicates don't care.

Offset: 0x1ac I Read/Write: R/W I Reset: 0x00000015 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx010101)

Bit	Reset	Description
5	ENABLE	TRIM_PWRSAVE: Enables power saving mode by clock gating the unused taps in delay chain Active low signal, 0 - power saving mode enabled - clock gating is enabled for unused trimmer taps - may affect tap delay 1 - no power saving - all the trimmer taps are not clock gated 0 = ENABLE 1 = DISABLE
4:3	VREF_925_MV	SEL_VREF_LEVEL: Selects Vref voltage level 0 = VREF_875_MV 1 = VREF_900_MV 2 = VREF_925_MV 3 = VREF_950_MV
2	0x1	SEL_VREG: Select voltage supply for delay chain present in both Trimmer and DLLPROD value: 0x0 ***Software should set this to 0x0 before accessing SD/eMMC. This setting makes IB trimmer delay independent of VDD_CORE*** 0 - selects regulated reference voltage for trimmer and DLL supply - default (recommended option for tunable SD/eMMC modes) 1 - selects VAUXC for trimmer and DLL supply and shut down BG+REG circuit (can be used in non-tunable modes for power saving) Power up time for BG+REG is ~3 us (worst case). Power down time for BG+REG is ~1us(worst case). If software wants to turn on/off BG+REG when SDMMC is idle, it has to take hit of 3us power on time. When SEL_VREG is toggled, both DLL and rx clock trimmer output could glitch irrespective of input clock state which could cause corresponding rx CMD and DATA FIFOs to go into bad state. Software should issue SW_RESET_DAT and SW_RESET_CMD to reset host FIFOs after BG <-> VAUXC switching. This would ensure error free data transfers from there on. Powering down BG would need 3 us turn ON time which may cause IOPS reduction, if software shuts down BG after every transfer and enables it on seeing new transfer request; it may not be possible to do dynamic shut down of BG without stalling new requests.
1	0x0	SEL_VREF: Select reference voltage for voltage regulator 0 - selects Bandgap Voltage Reference (recommended option for SD/eMMC tunable modes) 1 - selects resistor divider voltage reference and power down bandgap Switching time between the supplies is 1us. When switching from one supply to other supply, we need to wait for at least 1us before doing any data transfers.
0	0x1	PD_BGREG: Not used - Dummy control Power down Band Gap voltage reference, voltage regulator and resistor chain voltage ref (BG+REG) present in custom I/O trimmer used for SD/eMMC bus tuning. Active High signal, PD_BGREG=1 => Power down BG+REG; PD_BGREG=0 => power up Power down and up time for BG+REG is 1us. If software wants to turn on/off BG+REG when SDMMC is idle, it has to follow 1us power on/off time. Back-up option for powering down BG. Software should clear this bit when it wants to turn ON BG by setting SEL_VREG=0. The original idea to have PD_BGREG pin is to provide power saving feature when the eMMC/SDMMC is in IDLE state, but not in DPD mode. This pin function is actually merged into SEL_VREG function -BG+REG circuit is shut-down when SEL_VREG=1 (trimmer powered by VAUXC)

32.9.4.13 SDMMC_VENDOR_TUNING_CNTRL0_0

Vendor Tuning Control0 Register

Offset: 0x1c0 I Read/Write: R/W I Reset: 0x74020090 (0bx11101000000001x0000000010010000) I Default: 0x00000040

Bit	Reset	SW Default	Description
30	0x1	_NONE_	RD_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on read data CRC error

Bit	Reset	SW Default	Description
29	0x1	_NONE_	WR_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on write CRC error
28	0x1	_NONE_	CMD_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on cmd CRC error
27	0x0	_NONE_	RETUNING_REQ_EN_ON_CRC_ERR_DETECTION: Re-tuning request is generated when set to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
26	0x1	_NONE_	TUNING_ERR_EN_ON_CRC_ERR_DETECTION: Tuning error is generated to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
25:18	0x0	_NONE_	START_TAP_VAL: start tap value to be used by tuning; start_tap should be multiple of step_size chosen and its valid range is 0-255; Not valid when TAP_VAL_UPDATED_BY_HW is set to zero. Tuning algorithm uses this as the start tap value for scanning through trimmer taps.
17	0x1	_NONE_	TAP_VAL_UPDATED_BY_HW: This bit is functional only in tunable modes (SDR50, SDR104 and HS200) software can choose to update the tap value by itself by clearing this bit; Preferred value is 1 - tap value is updated by hardware. If this bit is cleared, hardware does not update tap_val per every tuning iteration. Software can update it as desired. Tuning pattern match is indicated by sampling_clock_select per every tuning iteration. Software has to maintain the status of each tuning iteration and determine the best PASS window to fix the final sampling point. Software can program NUM_TUNING_ITERATIONS as desired. Once the number of tuning commands issued reaches number of tuning iterations programmed, execute_tuning bit will be cleared to indicate the completion of tuning procedure. Note that using this option violates Host specification but provided for legacy reasons. It helps us in using legacy software tuning solution in case hardware solution does not work.
15:13	TRIES_40	_NONE_	NUM_TUNING_ITERATIONS: The number of tuning iterations to be used by tuning circuit; Preferred value is 40 iterations as per Host specification. Using other values violates the specification but provided for trimmer characterization purpose. 0 = TRIES_40 1 = TRIES_64 2 = TRIES_128 3 = TRIES_192 4 = TRIES_256
12:6	0x2	0x1	MUL_M: implements a multiplier - M+1 Final tap value is derived from best passing window and calculated as follows. Final tap value = first_pass + ((last_pass - first_pass)*Q); where Q = percentage of pass window;default-75% Q = M+1/(2^N); N:1...7 M: should be in range [0:2^N-1];
5:3	0x2	_NONE_	DIV_N: implements a divider - 2^N; maximum div is 2^7 =>128
2:0	0x0	_NONE_	TUNING_WORD_SEL: Selects desired word from 256-bit tuning status bitmap status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.4.14 SDMMC_VENDOR_TUNING_CNTRL1_0

Vendor Tuning Control1 register

Different step size is required in SDR50 mode to cover two UI (100MHz => 2*10ns)

With 70ps/tap trimmer resolution, we can cover almost 2UI using step_size=8 in SDR50 and step_size=4 in SDR104.

Tuning will be done in HS200 - SDR mode only.

HS200 - tuning at 200 MHz - UHS_SEL should be SDR104 for executing tuning

Before initiating data transfers in HS400 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 200 MHz

For overclocked modes: Before initiating data transfers in HS667 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 333MHz

> ***Note:*** *eMMC bus tuning will always be run in HS200 mode only.*

Use step_size = 1 to scan through all taps due to hardware issue.

Offset: 0x1c4 I Read/Write: R/W I Reset: 0x00000023 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010x011) I Default: 0x00000000

Bit	Reset	SW Default	Description
6:4	0x2	0x0	STEP_SIZE_SDR104_HS200: tap_val is incremented by step_size for every tuning iteration - used in SDR104/ HS200/HS400 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD For overclocking modes, set this field to 0x1
2:0	0x3	0x0	STEP_SIZE_SDR50: tap_val is incremented by step_size for every tuning iteration - used in SDR50 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD

32.9.4.15 SDMMC_VENDOR_TUNING_STATUS0_0

Vendor Tuning Status0 register

Offset: 0x1c8 I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	STATUS_WORD: Each bit indicates the status of each tuning iteration, when tap value is updated by hardware (TAP_VAL_UPDATED_BY_HW=1); 0-Tuning pattern not matched 1-tuning pattern matched We have a total of 256 tap values. Software can issue a maximum.of 256 tuning commands for debug. Software needs to read this register eight times to get status of all 256 iterations by changing tuning_word_sel status[255:0] is left shifted and loaded into this register every time when tuning_word_sel is changed status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.4.16 SDMMC_VENDOR_TUNING_STATUS1_0

Vendor Tuning Status1 register

Offset: 0x1cc I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	PASS_WINDOW_END_BEFORE_FINE_TUNING: End tap value of best PASS window found by scan FSM
23:16	X	PASS_WINDOW_START_BEFORE_FINE_TUNING: Start tap value of best PASS window found by scan FSM
15:8	X	PASS_WINDOW_END_AFTER_FINE_TUNING: End tap value of best PASS window after fine tuning
7:0	X	PASS_WINDOW_START_AFTER_FINE_TUNING: Start tap value of best PASS window after fine tuning

32.9.4.17 SDMMC_VENDOR_CLK_GATE_HYSTERESIS_COUNT_0

Vendor Clk gating Hysteresis Counter initial value

Offset: 0x1d0 I Read/Write: R/W I Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx001111)

Bit	Reset	Description
5:0	0xf	CLK_COUNT: Before gating second level clocks controller will wait for these many cycles. we can recover if any idle windows are missed in clken equation

32.9.4.18 SDMMC_VENDOR_PRESET_VAL0_0

Vendor Preset Value Registers

SD host specification defines one preset value register for each bus speed mode which should be set by host by some unique method.

Preset values vary based on the base frequency used which is in software (Tegra system driver) control.

Tegra System driver supposed to set BASE_CLK_FREQ in VENDOR_CLOCK_CNTRL register before handling over the control to SD host standard driver.

In the similar way, system driver should set below vendor preset values based on the base clock frequency and the desired card clock frequency in each bus speed mode

This should be done after every time SDMMC is reset and after every soft reset.

This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.

Default values are set assuming base clock frequency=208MHz.

Offset: 0x1d4 I Read/Write: R/W I Reset: 0x00201000 (0bxx00000000100000000010000000000000)

Bit	Reset	Description
29:20	0x2	SDCLK_FREQ_SEL_HIGH_SPEED: System software programs 10-bit divider value to generate SDCLK in default speed mode (<50 MHz,3.3V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
19:10	0x4	SDCLK_FREQ_SEL_DEFAULT: System software programs 10-bit divider value to generate SDCLK in default speed mode (<25MHz,3.3V signaling). This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x4 assuming 208 MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_INIT: System software programs 10-bit divider value to generate desired SDCLK frequency during initialization. This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_LOW register field. For example, if 400 kHz SDCLK is desired at base clk freq=48 MHz, this register should be programmed with 0x3C

32.9.4.19 SDMMC_VENDOR_PRESET_VAL1_0

Offset: 0x1d8 I Read/Write: R/W I Reset: 0x00100804 (0bxx00000000100000000100000000100)

Bit	Reset	Description
29:20	0x1	SDCLK_FREQ_SEL_SDR50: System software programs 10-bit divider value to generate SDCLK in SDR50 mode (<100 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x1 (gives 2N divider) assuming 208 MHz base clock
19:10	0x2	SDCLK_FREQ_SEL_SDR25: System software programs 10-bit divider value to generate SDCLK in SDR25 mode (<50 MHz, 1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
9:0	0x4	SDCLK_FREQ_SEL_SDR12: System software programs 10-bit divider value to generate SDCLK in SDR12 mode (<25 MHz, 1.8V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_HIGH register field

32.9.4.20 SDMMC_VENDOR_PRESET_VAL2_0

Offset: 0x1dc I Read/Write: R/W I Reset: 0x00000800 (0bxxxxxxxxxxxxx00000000100000000000)

Bit	Reset	Description
19:10	0x2	SDCLK_FREQ_SEL_DDR50: System software programs 10-bit divider value to generate SDCLK in DDR50 mode (<50 MHz, 1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x2 assuming 208 MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_SDR104: System software programs 10-bit divider value to generate SDCLK in SDR104 mode (<208 MHz, 1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_LOW register field

32.9.4.21 SDMMC_SDMEMCOMPPADCTRL_0

SDMEMCOMP Pad control register

This register is used to control COMP pad inputs.

Offset: 0x1e0 I Read/Write: R/W I Reset: 0x88000001 (0b10x01xxxxxxxxxxxxxxx0xxx0000001) I Default: 0x00000000

Bit	Reset	SW Default	Description
31	0x1	0x0	PAD_E_INPUT_OR_E_PWRD: used to control E_INPUT (for SDMMC1/3) and E_PWRD (for SDMMC2/4) input of pu/pd comp pad should be set before starting auto-cal and cleared once auto-calibration is done (for power saving). NOTE: E_PWRD = !PAD_E_INPUT_OR_E_PWRD and E_INPUT = PAD_E_INPUT_OR_E_PWRD
30	0x0	_NONE_	COMP_PAD_REG_ON: If set, turns ON regulator in comp pad. Software should not set this bit.
28:27	0x1	_NONE_	COMP_PAD_DRV_TYPE: used to control drv_type input of BDSDMEMLVCOMP_C pad
10	0x0	_NONE_	COMP_PAD_E_TEST_OUT: used to control e_test_out input of COMP pad
6:4	0x0	_NONE_	COMP_PAD_TEST_SEL: used to control test_sel input of COMP pad
3:0	0x1	_NONE_	SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL: used to control vref_sel input of COMP pad. Software should set this to 0x7.

32.9.4.22 SDMMC_AUTO_CAL_CONFIG_0

SDMEMCOMP pad auto-calibration settings

AUTO_CAL_SLW_OVERRIDE

0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output

DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3]

1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs

AUTO_CAL_OVERRIDE

0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting

1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly

Offset: 0x1e4 I Read/Write: R/W I Reset: 0x00010000 (0b0000xxxxxxxxx001x0000000x0000000) I Default: 0x20000000

Bit	Reset	SW Default	Description
31	0x0	_NONE_	AUTO_CAL_START: Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	0x0	_NONE_	AUTO_CAL_OVERRIDE: AUTOCAL override. 0 = NORMAL: 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 = OVERRIDE: 1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly
29	DISABLED	0x1	AUTO_CAL_ENABLE: AUTOCAL enable. 0 = DISABLED: 0 (disabled): use sdmmc2tmc_cfg* register settings for pullup/dn 1 = ENABLED: 1 (normal operation): use SDMMC generated pullup/dn (override or AUTOCAL)
28	0x0	_NONE_	AUTO_CAL_SLW_OVERRIDE: AUTOCAL slew rate override 0 = NORMAL: 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3] 1 = OVERRIDE: 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs
18:16	0x1	_NONE_	AUTO_CAL_STEP: calibration step interval (in microseconds)
14:8	0x0	_NONE_	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
6:0	0x0	_NONE_	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

32.9.4.23 SDMMC_AUTO_CAL_INTERVAL_0

SDMEMCOMP pad calibration interval - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1e8 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	AUTO_CAL_INTERVAL: 0: do calibration once Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.This feature is not supported from Tegra X1 due to movement of level shifters to COMP pad.Software should trigger calibration at regular intervals, if needed.

32.9.4.24 SDMMC_AUTO_CAL_STATUS_0

SDMEMCOMP pad calibration status - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1ec I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (AUTO_CAL_ACTIVE == 0)
30:24	X	AUTO_CAL_PULLDOWN_ADJ: Pulldown code sent to pads
22:16	X	AUTO_CAL_PULLUP_ADJ: Pullup code sent to pads
14:8	X	AUTO_CAL_PULLDOWN: Pulldown code generated by auto-calibration
6:0	X	AUTO_CAL_PULLUP: Pullup code generated by auto-calibration

32.9.4.25 SDMMC_IO_SPARE_0

SPARE bits. These SPARE_OUT bits go to pipe -> pad and then come back as SPARE_IN

SPARE_OUT[3]: IO_SPARE[19] - used as MUX select which selects between one cycle delay and two cycle delay versions of cmd_oen to mask wdata of IB capture flop.

0x0: selects two cycle delayed version

0x1: selects one cycle delayed version - recommended

SPARE_OUT[2]: IO_SPARE[18] - used as active low enable for gating both CMD_IN and DAT_IN async FIFOs wdata when we are driving CMD/DAT lines.

0x0: write 1 when OEN is active and Zi value when OEN is not active into FIFO (default)

0x1: write Zi value into FIFO irrespective of OEN state.

Offset: 0x1f0 I Read/Write: R/W I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx) I Default: 0x00080008

Bit	R/W	Reset	SW Default	Description
31:16	RW	X	0x8	SPARE_OUT:PROD value: Software should set SPARE_OUT[3]: IO_SPARE[19] to 1 before accessing SD/eMMC.
15:0	RO	X	0x8	SPARE_IN

32.9.4.26 SDMMC_SDMMC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register.

> ***Note:*** *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility*

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and mc side of the mccif. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clk is enabled.

With wclk/rclk override mode = ON, the clock is always on inside mccif and PC.

A '1' written to the cclk override field keeps client's clk always on inside mccif.

Offset: 0x1f4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	0x0	SDMMCA_RCLK_OVR_MODE
19	0x0	SDMMCA_WCLK_OVR_MODE
18	0x0	SDMMCA_CCLK_OVERRIDE
17	0x0	SDMMCA_RCLK_OVERRIDE
16	0x0	SDMMCA_WCLK_OVERRIDE

32.9.4.27 SDMMC_TIMEOUT_WCOAL_SDMMC_0

Write Coalescing Time-Out Register

Note: *Write coalescing is no longer supported by the MCCIF clients.*

Registers are retained for software compatibility but are not used by the hardware.

Offset: 0x1f8 I Read/Write: R/W I Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	SDMMCWA_WCOAL_TMVAL

32.9.5 SDMMC4 Vendor Registers

32.9.5.1 SDMMCAB_VENDOR_CLOCK_CNTRL_0

The following registers are Vendor Specific Registers and are mapped to Vendor Specific Address Space (0x100 - 0x1FF)

Vendor Clock Control Register

Offset: 0x100 I Read/Write: R/W I Reset: 0x0000d02d (0bxx00000000000011010000x0101101)

Bit	Reset	Description
29	0x0	DIFF_CLK_SEL: If set, selects differential CLK and DQS; Used by eMMC IOBRICK only. Software should set this appropriately based on eMMC part used. E_INPUT_* (CMD/DAT/CLK) of emmc IOBRICK should be set to 0 when differential signaling is used. default is '0' - selects single ended signaling for clk/dqs
28:24	0x0	TRIM_VAL: Tap value for output data path trimmer This determines the trimmer value needed to drive the output data correctly. The tap for outbound trimmer is single MUX. The trim settings required are within very small (0-3) with absolute delay requirement of ~400ps. Minimal change with PVT variations. Following values are recommended. These are common for all speed modes of SD/eMMC. These are the initial values to start with. SDMMC1 - 2 SDMMC2 - 8 SDMMC3 - 3 SDMMC4 - 8

Bit	Reset	Description
23:16	0x0	TAP_VAL: Tap value for input data path trimmer This determines the tap value needed to sample the input data correctly. Delay per each tap can range from 70ps (hv_ff) to 505ps (lv_ss). Following values are recommended. These are the initial values to start with. 1> SDR12/SDR25/High Speed DDR modes: If I/O pad trimmer is used (default): SDMMC1 - 4 SDMMC2 - 0 SDMMC3 - 3 SDMMC4 - 0 If core legacy trimmer is used: SDMMC1 - 3 SDMMC2 - 3 SDMMC3 - 3 SDMMC4 - 3 2> SDR50 and SDR104/HS200/HS400/HS667 should use tuning procedure to determine the tap value
15:8	0xd0	BASE_CLK_FREQ: System software programs the base SD/MMC clock frequency into this register before loading the SD/MMC driver.This should not be touched when software wants to use SD card in uhs-II mode. This value is readable in the standard, but not writeable, SDMMC_CAPABILITIES_0_BASE_CLOCK_FREQUENCY register field. Software should program the actual clock frequency programmed in CAR registers CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC*_0 for each SDMMC controller. This should be done after every time SDMMC is reset and after every soft reset. This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.
6	0x0	LEGACY_CLKEN_OVERRIDE: Override for sdmmc_legacy_g_clk clken; Set this to 0 when in UHS-II mode to save power 0 = NORMAL: 0 -> sdmmc_legacy_g_clk is gated 1 = OVERRIDE: 1 -> sdmmc_legacy_g_clk is not gated
5	0x1	SDR50_TUNING_OVERRIDE: override the SDR50_TUNING capabilities bit. Software should only set this bit if it is required to use Tuning for SDR50. (only supported for SDMMC1 and SDMMC3) 0 = NORMAL: 0 -> No Tuning support advertised for SDR50 mode. 1 = OVERRIDE: 1 -> Tuning support is enabled for SDR50 mode.
4	0x0	UHS2_CAPABILITY_OVERRIDE: override the UHS-II capabilities bit. 0 = NORMAL: 0 -> UHS-II support is unadvertised 1 = OVERRIDE: 1 -> UHS-II support is advertised
3	0x1	PADPIPE_CLKEN_OVERRIDE: Override for padmacro and pipemacro clken. 0 = NORMAL: 0 -> CLKEN is de-asserted when internal CLKEN is de-asserted. 1 = OVERRIDE: 1 -> CLKEN is kept asserted even when internal CLKEN is de-asserted. Software should always program this to 0x1 (OVERRIDE).
2	0x1	SPI_MODE_CLKEN_OVERRIDE: Override for CLKEN during SPI_MODE during sw_reset. 0 = NORMAL: 0 -> CLKEN is de-asserted while doing sw_reset. 1 = OVERRIDE: 1 -> CLKEN is kept asserted while doing sw_reset. Software should always program this to 0 (NORMAL).
1	0x0	INPUT_IO_CLK: Feedback clock is selected by default. Software should not change this. Disabling Feedback clock will select Internal Clock that requires different TAP Value Programming. 0 = FEEDBACK 1 = INTERNAL
0	0x1	SDMMC_CLK: This is set when sdmmc_clk is supplied by the CAR module.Prior to sdmmc_clk switch OFF.This bit should be written '0'. By writing zero,the asynchronous card interrupt is routed to the Interrupt controller. 0 = DISABLE 1 = ENABLE

32.9.5.2 SDMMCAB_VENDOR_SYS_SW_CNTRL_0

Vendor System Software Control Register

Offset: 0x104 I Read/Write: R/W I Reset: 0x38600002 (0b00111000011xxxxxx0000000x0000010)

Bit	Reset	Description
31	0x0	ENHANCED_STROBE_MODE: Enables enhanced strobe mode in HS400/HS667 mode for eMMC5.x devices 0 - cmd_in(Resp) is sampled by loopback clock which requires tuning 1 - cmd_in(Resp) is sampled by DQS_in which requires no tuning software has to set this bit appropriately based on device capability since this is an optional feature for eMMC5.x devices. If device supports this feature, software should set this bit to avoid tuning.
30	0x0	USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT: When set, uses TMCLK data timeout counter for generating wr_crc_status data-timeout When cleared, uses sdmmc_clk for maintaining wr_crc_status data timeout counter
29	0x1	USE_NCRC_FOR_WR_CRC_STATUS_TIMEOUT_VAL:This field is valid only when USE_TMCLK_FOR_WR_CRC_STATUS_TIMEOUT is set to 0. When cleared, uses data timeout value as wr CRC status timeout value (specification defined one) When set, uses Ncrc cycles as timeout value
28	0x1	USE_TMCLK_FOR_DATA_TIMEOUT: When set, uses TMCLK data timeout counter for generating legacy data timeout error (except wr_crc_status timeout) When cleared, uses sdmmc_clk for maintaining data timeout counter
27:24	0x8	DEVICE_BUSY_WAIT_CYCLES:In HS400/667 mode, busy should be sampled using clock instead DQS.So, Dat0 from card is passed to two different async FIFOs (one is running in sdmmc_clk and other is in sdmmc_dqs).CRC status is taken from dat0 sampled in DQS and busy should be checked after CRC status end bit reception using dat0 sampled in clk.Both clk and DQS dat0 paths don't have same prop delay due to sync-ers used in async FIFOs and also due to phase difference between clk and dqs.Due to this delay difference between the paths, we may sample wrong busy status if we sample busy immediately after receiving END bit of CRC status. To avoid incorrect sampling of busy, a wait state before busy polling is added. This register field is used to load wait_cycles counter. Note that this counter is used only in HS400/667 mode.
23	0x0	ALLOW_CARD_CLK_STALLS_IN_WR: When set, allows card clock stopping during transfer of data within a block in HIGH SPEED DDR/HS400 writes.
22	0x1	EMMC_IOBRICK_CLK_DATA:Used to drive AP_CLK and AN_CLK input of iobrick.0x1 - clk_out will be same as iobrick_clk_in0x0 - clk_out will be inverted iobrick_clk_in
21	0x1	QUALIFY_WITH_RD_DATA_VLD: There is async FIFOs in both cmd_in and dat_in paths in padmacro which are used in tunable modes. When this bit set, rdata from FIFO is treated as valid data only when rd_req is high. This is needed to handle bubbles on 'rd_req' when MTBF is high.
14	0x0	SD_BUS_POWER_ON_OFF_INT_STATUS: SD_BUS_POWER was changed. System software can use this interrupt to implement power switch.
13	0x0	VOLT_SWITCH_INT_STATUS: VOLT_18_EN was changed. System software can use this interrupt to implement a UHS-I voltage switch procedure for a standard SD Host driver 0 = NO_INT 1 = GEN_INT
12	0x0	TUNING_SYS_INT_STATUS: CMD19 was issued while EXECUTE_TUNING was set. System software can use this interrupt to implement a UHS-I tuning procedure for a standard SD Host driver. 0 = NO_INT 1 = GEN_INT
11:8	0x0	TUNING_ASYNC_FIFO_ADDNL_DELAY: In tunable modes, data/cmd IB path uses async FIFO which contributes additional delay to cmd/resp and wdata/CRC token round trip delay. Controller waits for this round trip delay before waiting for Ncr or Ncrc. This is required not to latch previous data/cmd driven by host itself as start bits of resp or CRC.Round trip delay is calculated for async FIFO with sync-2d. This register field holds the additional delay in cycles which should be added to round trip delay. This additional delay is caused by increasing sync-er depth (more than 2) or addition of output flop stage in async FIFO. Default value is zero. NOTE: Software should not update this field unless a new PROD setting is given.
6	0x0	SD_BUS_POWER_ON_OFF_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when SD_BUS_POWER is changed. 0 = DISABLE 1 = ENABLE
5	0x0	VOLT_SWITCH_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when VOLT_18_EN is changed. 0 = DISABLE 1 = ENABLE
4	0x0	TUNING_SYS_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when CMD19 is issued while EXECUTE_TUNING is set. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x0	ASSERT_BUFF_RD_RDY_INT: Write a 1 to this field to assert sdmmc_interrupt_status_0_buffer_read_ready. Used by the system software that implements the tuning procedure to signal to the standard SD driver that the tuning process has completed 0 = DISABLE 1 = ENABLE
2	0x0	IO_TRIM_BYPASS: Override bit for selecting between core trimmer (Vcore dependent) and I/O trimmer (custom trimmer) in IB clock path. Default option is I/O trimmer; software should not set this field.
1	0x1	INT_MASK_WHILE_TUNING: As per specification, Host should not generate any interrupts (including cmd_complete and data_xfer_complete) except buffer_read_ready interrupt during tuning sequence is being performed. Software can override this behavior by clearing this bit - but this leads to a specification violation 0 = DISABLE 1 = ENABLE
0	0x0	SPI_MODE: This is a mirror bit.The SPI mode is set if this bit is set or CMD_XFER_MODE[7] is set Writing 1 will drive the CS Low and writing zero will de-assert the CS Signal 0 = DISABLE: 0 -> SPI mode is disabled. 1 = ENABLE: 1 -> SPI mode is enabled.

32.9.5.3 SDMMCAB_VENDOR_ERR_INTR_STATUS_0

Legacy Interrupt Status Register

The fields are valid when an error interrupt has occurred.

Offset: 0x108 I Read/Write: RO I Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	SDMMC_LEGACY_CTLR_IDLE: indicates legacy SD interface controller is idle - no active data transfers on legacy SD interface
17	X	READ_DATA_TIMEOUT: valid when a data timeout error occurs
16	X	WRITE_CRC_STATUS_TIMEOUT: valid when a data timeout error occurs
15	X	WRITE_BUSY_TIMEOUT: valid when a data timeout error occurs
14	X	RESP_BUSY_TIMEOUT: valid when a data timeout error occurs
13	X	SPI_WRITE_BUSY_TIMEOUT
12	X	SPI_RX_START_TOKEN_TIMEOUT
8:5	X	SPI_DAT_ERR_TOKEN: Data Error Token, while read from card.
4:0	X	SPI_DAT_RESPONSE: Data Response while write to card 5 = DATA_ACCEPTED 11 = CRC_ERR 13 = WRITE_ERR

32.9.5.4 SDMMCAB_VENDOR_CAP_OVERRIDES_0

Capabilities override bits

Offset: 0x10c I Read/Write: R/W I Reset: 0x00000007 (0bxxx0xxxxxxxxxxxxxx000000xxxx0111)

Bit	Reset	Description
28	0x0	CMD_QUEUING_MODE_EN: useful for eMMC5.x devices which support CMD queuing. Software should enable this when it wants to use cmd queuing of eMMC5.x devices in HS400 mode.
13:8	0x0	DQS_TRIM_VAL:PROD_VAL=0x11 - Tap value for incoming DQS path trimmer - used in HS400/HS667 modes
3	0x0	DRV_LPBK_CLK_ON_CMD_LINE: Loopback trimmed clock will be driven onto cmd line, if this bit set to 1. Should be set to zero during normal data transfers.Useful in debug.
2	0x1	VOLTAGE_3_3_V_SUPPORT_OVERRIDE: Voltage support 3_3_V override
1	0x1	VOLTAGE_3_0_V_SUPPORT_OVERRIDE: Voltage support 3_0_V override
0	0x1	VOLTAGE_1_8_V_SUPPORT_OVERRIDE: Voltage support 1_8_V override

32.9.5.5 SDMMCAB_VENDOR_BOOT_CNTRL_0

Vendor Boot Control Register

This register is used to configure Boot Mode to support MMC v4.3 cards.

Offset: 0x110 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BOOT_ACK: This bit is used to support Boot Option in MMC 4.3 version cards. If set Boot acknowledgment is given by card else not given by card 0 = DISABLE 1 = ENABLE
0	0x0	BOOT: This bit enables/disable BootOption1.If set BootOption1 is enabled, hardware auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

32.9.5.6 SDMMCAB_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 0x114 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles. Boot Acknowledgment Timeout error occurs(VENDOR_SPECIFIC_ERR[0])

32.9.5.7 SDMMCAB_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 0x118 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxx000000000000000000000000)

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles. Then Data Timeout error occurs.

32.9.5.8 SDMMCAB_VENDOR_DEBOUNCE_COUNT_0

Debounce Counter Value Register

The Debounce Counter runs on 32 kHz clock. Keeping the default value to 100 ms = (100 * 32 cycles/1 ms) = 3200 cycles for 100 ms = 0xC80

Offset: 0x11c I Read/Write: R/W I Reset: 0x00000c80 (0bxxxxxxxx000000000000110010000000)

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32KHz clock cycles is programmed to meet Debounce period of the card slot.

32.9.5.9 SDMMCAB_VENDOR_MISC_CNTRL_0

Miscellaneous Vendor Control Register

SDMMC_SPARE0: Spare register bits with reset value of 0

SDMMC_SPARE0[0]: SW_RESET_CLKEN_OVERRIDE, override the sdmmc_clken when doing SW_RESET if set to 1.

SDMMC_SPARE0[1]: When set, allows SD clock to be stopped in the middle of a read data block while in SDR104/ HS400/HS667 modes (allow_sdr104_intrablock_stalls).

Unsafe for at least some SD/eMMC cards, but may improve SDR104 DMA read performance in some cases.

SDMMC_SPARE0[2]: When set, SDR104 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR104

SDMMC_SPARE0[3]: When set, SDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR50

SDMMC_SPARE0[5]: When 0, masks the pad macro's "high speed" enable to 0, causing the pad macro to always launch data on the falling edge of the clock. This prevents the SD Host driver's setting of

SDMMC_POWER_CONTROL_HOST_x_HIGH_SPEED_EN from undesirably affecting the output timing.

SDMMC_SPARE0[7:6]: Number of pipe stages.

SDMMC_SPARE0[8]: When set, DDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_DDR50

SDMMC_SPARE1: Spare register bits with reset value of 1

SDMMC_SPARE1[0]: CMD_CONFLICT_DISABLE, disable command line conflict if set to 1.

SDMMC_SPARE1[1]: Control bit to select between internal and external loop back clock. 1-external 0-internal

Tegra X1 does not support external loopback for any SDMMC controller.

Offset: 0x120 I Read/Write: R/W I Reset: 0xffff0098 (0b11111111111111110000000010011000)

Bit	Reset	Description
31:16	0xffff	SDMMC_SPARE1: Spare register bits with reset value of 1
15:1	0x4c	SDMMC_SPARE0: Spare register bits with reset value of 0x40
0	0x0	ERASE_TIMEOUT_LIMIT: Erase timeout value. 0 = FINITE: Finite, It is limited to the programmed value in the DATA_TIMEOUT_VALUE 1 = INFINITE: Infinite, Controller would be monitoring until the card is busy.

32.9.5.10 SDMMCAB_MAX_CURRENT_OVERRIDE_0

Offset: 0x124 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	OVERRIDE_FOR_1_8V: Maximum override for 1.8V VDD1
15:8	0x0	OVERRIDE_FOR_3_0V: Maximum override for 3.0V VDD1
7:0	0x0	OVERRIDE_FOR_3_3V: Maximum override for 3.3V VDD1

32.9.5.11 SDMMCAB_MAX_CURRENT_OVERRIDE_HI_0

Offset: 0x128 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	OVERRIDE_FOR_1_8V_VDD2: Maximum override for 1.8V VDD2

32.9.5.12 SDMMCAB_VENDOR_IO_TRIM_CNTRL_0

Vendor I/O trimmer control register

Used to configure I/O trimmer

Table 210: Truth Table

Input Pins			Output	Comments
E_DPD	SEL_VREG	SEL_VREF	CLKOUT	
1	x[1]	x	0	The cell is in deep power down mode.
0	1	x	based on ip_clk_select selected clock input	Trimmer is powered by VAUXC.
0	0	x	based on ip_clk_select selected clock input	Trimmer is powered by regulated voltage.

1. 'x' indicates don't care.

Offset: 0x1ac I Read/Write: R/W I Reset: 0x00000015 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx010101)

Bit	Reset	Description
5	ENABLE	TRIM_PWRSAVE: Enables power saving mode by clock gating the unused taps in delay chain Active low signal, 0 - power saving mode enabled - clock gating is enabled for unused trimmer taps - may affect tap delay 1 - no power saving - all the trimmer taps are not clock gated 0 = ENABLE 1 = DISABLE
4:3	VREF_925_MV	SEL_VREF_LEVEL: Selects Vref voltage level 0 = VREF_875_MV 1 = VREF_900_MV 2 = VREF_925_MV 3 = VREF_950_MV
2	0x1	SEL_VREG: Select voltage supply for delay chain present in both Trimmer and DLLPROD value: 0x0 ***Software should set this to 0x0 before accessing SD/eMMC. This setting makes IB trimmer delay independent of VDD_CORE*** 0 - selects regulated reference voltage for trimmer and DLL supply - default (recommended option for tunable SD/eMMC modes) 1 - selects VAUXC for trimmer and DLL supply and shut down BG+REG circuit (can be used in non-tunable modes for power saving) Power up time for BG+REG is ~3us(worst case). Power down time for BG+REG is ~1us(worst case). If software wants to turn on/off BG+REG when SDMMC is idle, it has to take hit of 3us power on time. When SEL_VREG is toggled, both DLL and rx clock trimmer output could glitch irrespective of input clock state which could cause corresponding rx CMD and DATA FIFOs to go into bad state. Software should issue SW_RESET_DAT and SW_RESET_CMD to reset host FIFOs after BG <-> VAUXC switching. This would ensure error free data transfers from there on. Powering down BG would need 3 us turn ON time which may cause IOPS reduction, if software shuts down BG after every transfer and enables it on seeing new transfer request; it may not be possible to do dynamic shut down of BG without stalling new requests.
1	0x0	SEL_VREF: Select reference voltage for voltage regulator 0 - selects Bandgap Voltage Reference (recommended option for SD/eMMC tunable modes) 1 - selects resistor divider voltage reference and power down bandgap Switching time between the supplies is 1 us. When switching from one supply to other supply, we need to wait for at least 1us before doing any data transfers.
0	0x1	PD_BGREG: Not used - Dummy control Power down Band Gap voltage reference, voltage regulator and resistor chain voltage ref (BG+REG) present in custom I/O trimmer used for SD/eMMC bus tuning. Active High signal, PD_BGREG=1 => Power down BG+REG; PD_BGREG=0 => power up Power down and up time for BG+REG is 1us. If software wants to turn on/off BG+REG when SDMMC is idle, it has to follow 1 us power on/off time. Back-up option for powering down BG. Software should clear this bit when it wants to turn ON BG by setting SEL_VREG=0. The original idea to have PD_BGREG pin is to provide power saving feature when the eMMC/SDMMC is in IDLE state, but not in DPD mode. This pin function is actually merged into SEL_VREG function -BG+REG circuit is shut-down when SEL_VREG=1 (trimmer powered by VAUXC)

32.9.5.13 SDMMCAB_VENDOR_DLLCAL_CFG_0

Vendor DLL calibration configuration register

This register is used to setup the DLL Calibration settings. Software has to run DLL calibration first before initiating data transfers in HS400 or HS667 modes and re-calibration is required whenever there is a change in eMMC device clock frequency in HS400/HS667 modes.

Offset: 0x1b0 I Read/Write: R/W I Reset: 0x16083504 (0b0x01011000001000001101010000010 0)

Bit	Reset	Description
31	0x0	CALIBRATE: CALIBRATE is used to start a DLL calibration process. Software should set this bit to trigger calibration and should not clear this bit. This bit will be cleared once the calibration process is completed.
29:25	0xb	END_COUNT: END_COUNT determines the end condition of the calibration. If USE_STATIC_CYCLES is not set, a usec timer is loaded with (2^END_COUNT) and calibration will be stopped once the timer expires. Recommended timer value is 1 ms. If USE_STATIC_CYCLES is set and if the trimmer tap value being calibrated has not changed for this number of loops, the calibration ends. The number iterations is (2 ^ END_COUNT), if END_COUNT != 0
24	0x0	USE_STATIC_CYCLES: When set, calibration will be stopped when num static cycles reaches END_COUNT else usec timer is used. Recommended option is use usec timer.
23	0x0	IGNORE_START_TRIM: IGNORE_START is used to ignore the START_TRM value. Instead the calibration starts at the current programmed DDLLCAL value. This is intended to be used if we ever support periodic DDLL calibration.
22:16	0x8	START_TRIM: START_TRM specifies the starting trimmer value to calibrate the with.
15:12	0x3	FILTER_BITS: FILTER_BITS is the LSB of the counter to use for updating the trimmer value. The new trimmer value is trim_old + (count >> FILTER_BITS) - (trim_old >> FILTER_BITS) FILTER_BITS chosen < bits used for SAMPLE_COUNT

Bit	Reset	Description
11:8	0x5	SAMPLE_COUNT: SAMPLE_COUNT is the number of times the phase detector is sampled before going onto the next set of trimmer values. The number of samples is (2 ^ SAMPLE_COUNT)
7:0	0x4	SAMPLE_DELAY: SAMPLE_DELAY is the number of sdmmc host clks from changing the trimmer value to when the phase detector is sampled.

32.9.5.14 SDMMCAB_VENDOR_DLL_CTRL0_0

Vendor DLL control register0

This register has software overrides for controlling both master and slave DLL inputs

Offset: 0x1b4 I Read/Write: R/W I Reset: 0x60000000 (0b01100000000000000000000000000000)

Bit	Reset	Description
31	0x0	MST_DLL_CLK_EN_OVERRIDE:Master DLL CLK_IN enable override 0 = NORMAL: 0 -> Mst DLL clk_in is gated when DLL calibration is not running 1 = OVERRIDE: 1 -> Mst DLL clk_in is not gated irrespective of DLL calibration status
30	0x1	TX_SLV_DLL_PWRSAVE:PWRSAVE is active low signal 1: no clock gating enabled for TX slave DLL 0: clock gating is enabled for unused delay taps in TX slv DLL to save power. This control does not affect the logic output of slave DLL; controls slv dll enable in iobrick
29	0x1	RX_SLV_DLL_PWRSAVE:PWRSAVE is active low signal 1: no clock gating is enabled for RX slave DLL0: clock gating is enabled for unused delay taps in RX slv DLL to save power. This control does not affect the logic output of slave DLL; controls slv dll enable in iobrick
28:22	0x0	TX_SLV_DLL_DLY_CODE:7-bit delay code (128 taps) to be applied to TX slave DLL when DLLCAL_BYPASS is enabled
21:15	0x0	RX_SLV_DLL_DLY_CODE:7-bit delay code (128 taps) to be applied to RX slave DLL when DLLCAL_BYPASS is enabled
14	DISABLED	DLLCAL_BYPASS:DLL calibration bypass enable 0 = DISABLED: 1: Programmed delay code will be applied to both master and slave DLLs; 1 = ENABLED: 0: Delay code determined by DLL calibration is applied to master and slave DLLS
13:7	0x0	TX_DLY_CODE_OFFSET: two-s complement offset will be added to delay code generated by calibration controller and sent to TX slv DLL
6:0	0x0	RX_DLY_CODE_OFFSET: two-s complement offset will be added to delay code generated by calibration controller and sent to RX slv DLL

32.9.5.15 SDMMCAB_VENDOR_DLL_CTRL1_0

Vendor DLL control register1

This register has software overrides for controlling both master and slave DLL inputs

Offset: 0x1b8 I Read/Write: R/W I Reset: 0x20208780 (0bx0100000x01000001000011110000000)

Bit	Reset	Description
30:24	0x20	REQD_DELAY_STEPS_TX: Delay required in steps of 1/64 UI - 0 to 63 steps. MST DLL is half cycle locked - UI=0.5 cycle=64 steps. Default value is quarter cycle. Calibrated code covers one UI of MST DLL. This reg field value is used to scale the delay code before sending it to TX slv DLL.
22:16	0x20	REQD_DELAY_STEPS_RX: Delay required in steps of 1/64 UI - 0 to 63 steps. MST DLL is half cycle locked - UI=0.5 cycle=64 steps. Default value is quarter cycle. Calibrated code covers one UI of MST DLL. This reg field value is used to scale the delay code before sending it to RX slv DLL.
15:11	0x10	MST_DLL_RESET_TIME: Master DLL reset duration - 16 cycles
10:7	0xf	SLV_DLL_SETTLE_TIME: Slave DLL requires 4 cycles settle time when dly code is changed for providing stable output
6:0	0x0	MST_DLL_DLY_CODE: 7-bit delay code (128 taps) to be applied to master DLL when DLLCAL_BYPASS is enabled

32.9.5.16 SDMMCAB_VENDOR_DLLCAL_CFG_STA_0

Vendor DLL calibration configuration register

This register is used to setup the DLL Calibration settings.

Software has to run DLL calibration first before initiating data transfers in HS400 or HS667 modes and re-calibration is required whenever there is a change in eMMC device clock frequency in HS400/HS667 modes. DLL calibration process is an open loop process in which DLL calibration controller can't decide the final tap value on its own by looking at PD output from DLL.

Summary of DLL calibration process implemented:

1. Software programs SAMPLE_COUNT (number of samples to be taken per every tap value) and STOP_TIMER_VALUE in microseconds.
2. These two values decide the stop condition of calibration.
3. DLL calibration is triggered on writing '1' into CALIBRATE register field by software. The controller sets CAL_ACTIVE bit to indicate calibration running status.
4. DLL controller starts calibration by applying a tap value (may use START_TAP programmed) to MST DLL and loads usec timer with STOP_TIMER_VALUE.
    a. After SAMPLE_DELAY (4 cycles of DLL in clock), it starts taking SAMPLE_COUNT number of PD samples. A phase counter is updated based on PD value per every cycle.
    b. Based on phase counter value, 'Filter' decides whether to increment or decrement current tap value to get next tap value to be applied in next iteration.
    c. FILTER_BITS decides the accuracy of the decision made. FILTER_BITS < bits used for SAMPLE_COUNT.
5. Repeat step 3 till STOP_TIMER expires.
6. The controller stops calibration (updating tap value) and latches the tap value used in last calibration iteration. CALIBRATE bit will be cleared at this step.
7. The controller adds offsets programmed to calibrated tap value and applies the final codes to SLAVE DLLs.
8. The controller waits for slave DLL settle time and clears CAL_ACTIVE register bit.
9. Software polls for CAL_ACTIVE=0 and can start data transfers.

DLL lock time is represented as:

Each tap test time = (SAMPLE_DELAY + SAMPLE_COUNT) * DLL_CAL_CLK_PERIOD

Worst case DLL locking time = STOP_TIMER_VALUE = (NUM_CAL_ITERATIONS * (SAMPLE_DELAY + SAMPLE_COUNT) * DLL_CAL_CLK_PERIOD)

Default values chosen:

Worst case DLL locking time=1 ms = 1000us = STOP_TIMER_VALUE.

SAMPLE_COUNT=32 samples per one tap;

SAMPLE_DELAY=4 cycles

Tap value is decided by PD and filter logic.

Notes:

- *Software needs to calibrate DLL before using eMMC5.0 HS400/HS667 mode (before tuning process). Re-calibration is required whenever eMMC I/O clock frequency is changed. This can be done as a part of standard clock frequency change sequence.*
- *DLL calibration would work at when I/O CLK is running at 200 MHz and above.*
- *For I/O CLK frequency is below 200 MHz, software should calculate delay code as per below equation:*
- *Delay Code = (Delay Code at 200 MHz) * 200 MHz / Target Frequency*
- *Under 85 MHz, DLL delay code may reach its maximum code (128 taps) for FF chips. In this case, software should use 128 taps.*

- *Software can run DLL calibration at 200MHz once at boot time, save those values, scale and use them whenever frequency is changed. This is needed if software wants to do DFS.*

Offset: 0x1bc I Read/Write: R/W I Reset: 0xX0XXXXXX (0bxx010000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RO	X	DLL_CAL_ACTIVE: Calibration process status. Software has to wait for DONE once calibrate bit is cleared and before starting data transfers on eMMC interface. 0 = DONE: 0: DLL calibration is completed - calibration engine is idle and slave DLLs are ready for normal operation 1 = RUNNING: 1: DLL calibration is still running
30	RO	X	DLL_PD: Master DLL Phase detector output
29	RW	0x0	MST_DLL_RST_OVERRIDE_EN: When set, master dll reset is controlled by programming MST_DLL_RST_ Else reset is generated by DLL controller
28	RW	0x1	MST_DLL_RST_: Master DLL reset - active low signal - used when DLL_RST_OVERRIDE_EN is set minimum reset duration is 16 sdhost clk cycles
27	RW	0x0	SLV_DLL_CLK_OUT_DIS_OVERRIDE_EN: When set, slave dll clk_out_dis is controlled by programming SLV_DLL_CLK_OUT_DIS Else clk_out_dis is controlled by DLL controller
26	RW	0x0	SLV_DLL_CLK_OUT_DIS:Slave DLL clock out disable - used when SLV_DLL_CLK_OUT_DIS_OVERRIDE_EN is set 0: Delay line clock normal output1: DelayLIne clock output gated and grounded to 0.should be set to 1 when software tries to update slv dll delay code should be cleared after slave dll settle time - 3cycles
25	RW	0x0	MST_DLL_PWRDN_OVERRIDE_EN: When set, master dll can be kept in power down mode by programming MST_DLL_PWRDN field. Else pwrdn is controlled by DLL controller
24	RW	0x0	MST_DLL_PWRDN: Master DLL power down enable - active high control - used when MST_DLL_PWRDN_OVERRIDE_EN is set 0: power up; 1 - power down. Software can keep MST DLL in power down mode once calibration is done.MST DLL should be powered up before triggering calibration. Controls dll_en in iobrick
22:16	RO	X	TX_DLY_CODE_ADJ: delay code sent to TX slave DLL after applying offset; Valid only when dll auto-calibration is used. (calib_dly_code * reqd_dly_steps/64) + offset
14:8	RO	X	RX_DLY_CODE_ADJ: delay code sent to RX slave DLL after applying offset; Valid only when dll auto-calibration is used. (calib_dly_code * reqd_dly_steps/64) + offset
6:0	RO	X	DLY_CODE: Holds delay code determined by calibration process which is sent to MST dll during calibration; Valid only when DLL_CAL_ACTIVE is set

32.9.5.17 SDMMCAB_VENDOR_TUNING_CNTRL0_0

Vendor Tuning Control0 Register

Offset: 0x1c0 I Read/Write: R/W I Reset: 0x74020090 (0bx11101000000001x0000000010010000) I Default: 0x00000040

Bit	Reset	SW Default	Description
30	0x1	_NONE_	RD_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on read data CRC error
29	0x1	_NONE_	WR_DATA_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on write CRC error
28	0x1	_NONE_	CMD_CRC_ERR_EN: If cleared, Re-tuning request/tuning error is not generated on cmd CRC error
27	0x0	_NONE_	RETUNING_REQ_EN_ON_CRC_ERR_DETECTION: Re-tuning request is generated when set to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
26	0x1	_NONE_	TUNING_ERR_EN_ON_CRC_ERR_DETECTION: Tuning error is generated to initiate re-tuning by software to compensate for temperature drift when any data or cmd CRC errors are detected in SDR50/SDR104/HS200 modes
25:18	0x0	_NONE_	START_TAP_VAL: start tap value to be used by tuning; start_tap should be multiple of step_size chosen and its valid range is 0-255; Not valid when TAP_VAL_UPDATED_BY_HW is set to zero. Tuning algorithm uses this as the start tap value for scanning through trimmer taps.

Bit	Reset	SW Default	Description
17	0x1	_NONE_	TAP_VAL_UPDATED_BY_HW: This bit is functional only in tunable modes (SDR50, SDR104 and HS200). Software can choose to update the tap value by itself by clearing this bit; Preferred value is 1 - tap value is updated by hardware. If this bit is cleared, hardware does not update tap_val per every tuning iteration. Software can update it as desired. Tuning pattern match is indicated by sampling_clock_select per every tuning iteration. Software has to maintain the status of each tuning iteration and determine the best PASS window to fix the final sampling point. Software can program NUM_TUNING_ITERATIONS as desired. Once the number of tuning commands issued reaches number of tuning iterations programmed, execute_tuning bit will be cleared to indicate the completion of tuning procedure. Note that using this option violates Host specification but provided for legacy reasons. It helps us in using legacy software tuning solution in case hardware solution does not work.
15:13	TRIES_40	_NONE_	NUM_TUNING_ITERATIONS: The number of tuning iterations to be used by tuning circuit; Preferred value is 40 iterations as per Host specification. Using other values violates the specification but provided for trimmer characterization purpose. 0 = TRIES_40 1 = TRIES_64 2 = TRIES_128 3 = TRIES_192 4 = TRIES_256
12:6	0x2	0x1	MUL_M: implements a multiplier - M+1 Final tap value is derived from best passing window and calculated as follows. Final tap value = first_pass + ((last_pass - first_pass)*Q); where Q = percentage of pass window; default-75% Q = M+1/(2^N); N:1...7 M: should be in range [0:2^N-1];
5:3	0x2	_NONE_	DIV_N: implements a divider - 2^N; maximum div is 2^7 =>128
2:0	0x0	_NONE_	TUNING_WORD_SEL: Selects desired word from 256-bit tuning status bitmap status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.5.18 SDMMCAB_VENDOR_TUNING_CNTRL1_0

Vendor Tuning Control1 register

Different step size is required in SDR50 mode to cover two UI (100MHz => 2*10 ns)

With 70ps/tap trimmer resolution, we can cover almost 2UI using step_size=8 in SDR50 and step_size=4 in SDR104.

Tuning will be done in HS200 - SDR mode only.

HS200 - tuning at 200 MHz - UHS_SEL should be SDR104 for executing tuning

Before initiating data transfers in HS400 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 200 MHz

For overclocked modes: Before initiating data transfers in HS667 mode, tuning procedure should be executed in HS200 mode with I/O clock running at 333 MHz

---

Note: *eMMC bus tuning will always be run in HS200 mode only.*

Use step_size = 1 to scan through all taps due to hardware issue.

Offset: 0x1c4 I Read/Write: R/W I Reset: 0x00000023 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx010x011) I Default: 0x00000000

Bit	Reset	SW Default	Description
6:4	0x2	0x0	STEP_SIZE_SDR104_HS200: tap_val is incremented by step_size for every tuning iteration - used in SDR104/ HS200/HS400 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD For overclocking modes, set this field to 0x1
2:0	0x3	0x0	STEP_SIZE_SDR50: tap_val is incremented by step_size for every tuning iteration - used in SDR50 mode increment = 2^step_size; step_size should be in range 0-4. Others are RSVD

32.9.5.19 SDMMCAB_VENDOR_TUNING_STATUS0_0

Vendor Tuning Status0 register

Offset: 0x1c8 I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	STATUS_WORD: Each bit indicates the status of each tuning iteration, when tap value is updated by hardware (TAP_VAL_UPDATED_BY_HW=1); 0-Tuning pattern not matched 1-tuning pattern matched We have a total of 256 tap values. Software can issue a maximum of 256 tuning commands for debug. Software needs to read this register eight times to get status of all 256 iterations by changing tuning_word_sel status[255:0] is left shifted and loaded into this register every time when tuning_word_sel is changed status_word[31:0] = status[255:0] >> (tuning_word_sel * 32)

32.9.5.20 SDMMCAB_VENDOR_TUNING_STATUS1_0

Vendor Tuning Status1 register

Offset: 0x1cc I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	PASS_WINDOW_END_BEFORE_FINE_TUNING: End tap value of best PASS window found by scan FSM
23:16	X	PASS_WINDOW_START_BEFORE_FINE_TUNING: Start tap value of best PASS window found by scan FSM
15:8	X	PASS_WINDOW_END_AFTER_FINE_TUNING: End tap value of best PASS window after fine tuning
7:0	X	PASS_WINDOW_START_AFTER_FINE_TUNING: Start tap value of best PASS window after fine tuning

32.9.5.21 SDMMCAB_VENDOR_CLK_GATE_HYSTERESIS_COUNT_0

Vendor Clk gating Hysteresis Counter initial value

Offset: 0x1d0 I Read/Write: R/W I Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx001111)

Bit	Reset	Description
5:0	0xf	CLK_COUNT: Before gating second level clocks controller will wait for these many cycles. we can recover if any idle windows are missed in clken equation

32.9.5.22 SDMMCAB_VENDOR_PRESET_VAL0_0

Vendor Preset Value Registers

SD host specification defines one preset value register for each bus speed mode which should be set by host by some unique method.

Preset values vary based on the base frequency used which is in software (Tegra system driver) control.

Tegra System driver supposed to set BASE_CLK_FREQ in VENDOR_CLOCK_CNTRL register before handling over the control to SD host standard driver.

In the similar way, system driver should set below vendor preset values based on the base clock frequency and the desired card clock frequency in each bus speed mode

This should be done after every time SDMMC is reset and after every soft reset.

This is important as all SDMMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.

Default values are set assuming base clock frequency=208 MHz.

Offset: 0x1d4 I Read/Write: R/W I Reset: 0x00201000 (0bxx00000000100000000100000000000)

Bit	Reset	Description
29:20	0x2	SDCLK_FREQ_SEL_HIGH_SPEED: System software programs 10-bit divider value to generate SDCLK in default speed mode (<50 MHz,3.3V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
19:10	0x4	SDCLK_FREQ_SEL_DEFAULT: System software programs 10-bit divider value to generate SDCLK in default speed mode (<25 MHz,3.3V signaling). This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x4 assuming 208 MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_INIT: System software programs 10-bit divider value to generate desired SDCLK frequency during initialization. This value is readable in the standard via PRESET_DEFAULT_AND_INIT_0_SDCLK_FREQ_VAL_LOW register field. For example, if 400 KHz SDCLK is desired at base clk freq=48 MHz, this register should be programmed with 0x3C

32.9.5.23 SDMMCAB_VENDOR_PRESET_VAL1_0

Offset: 0x1d8 I Read/Write: R/W I Reset: 0x00100804 (0bxx00000000100000000100000000100)

Bit	Reset	Description
29:20	0x1	SDCLK_FREQ_SEL_SDR50: System software programs 10-bit divider value to generate SDCLK in SDR50 mode (<100 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x1 (gives 2N divider) assuming 208 MHz base clock
19:10	0x2	SDCLK_FREQ_SEL_SDR25: System software programs 10-bit divider value to generate SDCLK in SDR25 mode (<50 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR50_AND_SDR25_0_SDCLK_FREQ_VAL_LOW register field. Default value is 0x2 assuming 208 MHz base clock
9:0	0x4	SDCLK_FREQ_SEL_SDR12: System software programs 10-bit divider value to generate SDCLK in SDR12 mode (<25 MHz,1.8V signaling). This value is readable in the standard via PRESET_SDR12_AND_HIGH_0_SDCLK_FREQ_VAL_HIGH register field

32.9.5.24 SDMMCAB_VENDOR_PRESET_VAL2_0

Offset: 0x1dc I Read/Write: R/W I Reset: 0x00000800 (0bxxxxxxxxxxxxx00000000100000000000)

Bit	Reset	Description
19:10	0x2	SDCLK_FREQ_SEL_DDR50: System software programs 10-bit divider value to generate SDCLK in DDR50 mode (<50 MHz,1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_HIGH register field. Default value is 0x2 assuming 208 MHz base clock
9:0	0x0	SDCLK_FREQ_SEL_SDR104: System software programs 10-bit divider value to generate SDCLK in SDR104 mode (<208 MHz,1.8V signaling). This value is readable in the standard via PRESET_DDR50_AND_SDR104_0_SDCLK_FREQ_VAL_LOW register field

32.9.5.25 SDMMCAB_SDMEMCOMPPADCTRL_0

SDMEMCOMP Pad control register

This register is used to control COMP pad inputs.

Offset: 0x1e0 I Read/Write: R/W I Reset: 0x88000001 (0b10x01xxxxxxxxxxxxxxxx0xxx0000001) I Default: 0x00000000

Bit	Reset	SW Default	Description
31	0x1	0x0	PAD_E_INPUT_OR_E_PWRD: used to control E_INPUT (for SDMMC1/3) and E_PWRD (for SDMMC2/4) input of pu/pd comp pad should be set before starting auto-cal and cleared once auto-calibration is done (for power saving) NOTE: E_PWRD = !PAD_E_INPUT_OR_E_PWRD and E_INPUT = PAD_E_INPUT_OR_E_PWRD
30	0x0	_NONE_	COMP_PAD_REG_ON: If set, turns ON regulator in comp pad. Software should not set this bit.
28:27	0x1	_NONE_	COMP_PAD_DRV_TYPE: used to control drv_type input of BDSDMEMLVCOMP_C pad
10	0x0	_NONE_	COMP_PAD_E_TEST_OUT: used to control e_test_out input of COMP pad

Bit	Reset	SW Default	Description
6:4	0x0	_NONE_	COMP_PAD_TEST_SEL: used to control test_sel input of COMP pad
3:0	0x1	_NONE_	SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL: used to control vref_sel input of COMP pad. Software should set this to 0x7.

32.9.5.26 SDMMCAB_AUTO_CAL_CONFIG_0

SDMEMCOMP pad auto-calibration settings

AUTO_CAL_SLW_OVERRIDE

0 (Normal operation): pad DRVDN/UP_SLWR/F tied to AUTO_CAL output

DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3]

1 (override): use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs

AUTO_CAL_OVERRIDE

0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting

1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly

Offset: 0x1e4 I Read/Write: R/W I Reset: 0x00010000 (0b0000xxxxxxxxx001x0000000x0000000) I Default: 0x20000000

Bit	Reset	SW Default	Description
31	0x0	_NONE_	AUTO_CAL_START: Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	0x0	_NONE_	AUTO_CAL_OVERRIDE: AUTOCAL override. 0 = NORMAL: 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 = OVERRIDE: 1 (override): use AUTO_CAL_PU/PD_OFFSET register values directly
29	DISABLED	0x1	AUTO_CAL_ENABLE: AUTOCAL enable. 0 = DISABLED: 0 (disabled): use sdmmc2tmc_cfg* register settings for pullup/dn 1 = ENABLED: 1 (normal operation): use SDMMC generated pullup/dn (override or AUTOCAL)
28	0x0	_NONE_	AUTO_CAL_SLW_OVERRIDE: AUTOCAL slew rate override 0 = NORMAL: 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3] 1 = OVERRIDE: 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs
18:16	0x1	_NONE_	AUTO_CAL_STEP: calibration step interval (in microseconds)
14:8	0x0	_NONE_	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
6:0	0x0	_NONE_	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

32.9.5.27 SDMMCAB_AUTO_CAL_INTERVAL_0

SDMEMCOMP pad calibration interval - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1e8 I Read/Write: R/W I Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	AUTO_CAL_INTERVAL: 0: do calibration once Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.This feature is not supported from Tegra X1 due to movement of level shifters to COMP pad.Software should trigger calibration at regular intervals, if needed.

32.9.5.28 SDMMCAB_AUTO_CAL_STATUS_0

SDMEMCOMP pad calibration status - Valid for SDMMC1/SDMMC3 Instances

Offset: 0x1ec I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (AUTO_CAL_ACTIVE == 0)
30:24	X	AUTO_CAL_PULLDOWN_ADJ: Pulldown code sent to pads
22:16	X	AUTO_CAL_PULLUP_ADJ: Pullup code sent to pads
14:8	X	AUTO_CAL_PULLDOWN: Pulldown code generated by auto-calibration
6:0	X	AUTO_CAL_PULLUP: Pullup code generated by auto-calibration

32.9.5.29 SDMMCAB_IO_SPARE_0

SPARE bits. These SPARE_OUT bits go to pipe -> pad and then come back as SPARE_IN

SPARE_OUT[3]: IO_SPARE[19] - used as MUX select which selects between one cycle delay and two cycle delay versions of cmd_oen to mask wdata of IB capture flop.

0x0: selects two cycle delayed version

0x1: selects one cycle delayed version - recommended

SPARE_OUT[2]: IO_SPARE[18] - used as active low enable for gating both CMD_IN and DAT_IN async FIFOs wdata when we are driving CMD/DAT lines.

0x0: write 1 when OEN is active and Zi value when OEN is not active into FIFO (default)

0x1: write Zi value into FIFO irrespective of OEN state.

Offset: 0x1f0 I Read/Write: R/W I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx) I Default: 0x00080008

Bit	R/W	Reset	SW Default	Description
31:16	RW	X	0x8	SPARE_OUT:PROD value: Software should set SPARE_OUT[3]: IO_SPARE[19] to 1 before accessing SD/eMMC.
15:0	RO	X	0x8	SPARE_IN

32.9.5.30 SDMMCAB_SDMMCAB_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register.

Note: *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility*

The clock override/ovr_mode fields of this register control the 2nd-level clock gating for the client and mc side of the mccif. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clk is enabled.

With wclk/rclk override mode = ON, the clock is always on inside mccif and PC.

A '1' written to the cclk override field keeps client's clk always on inside mccif.

Offset: 0x1f4 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	0x0	SDMMCA_RCLK_OVR_MODE
19	0x0	SDMMCA_WCLK_OVR_MODE
18	0x0	SDMMCA_CCLK_OVERRIDE

Bit	Reset	Description
17	0x0	SDMMCA_RCLK_OVERRIDE
16	0x0	SDMMCA_WCLK_OVERRIDE

32.9.5.31 SDMMCAB_TIMEOUT_WCOAL_SDMMCAB_0

Write Coalescing Time-Out Register

Note: *Write coalescing is no longer supported by the MCCIF clients.*

Registers are retained for software compatibility but are not used by the hardware.

Offset: 0x1f8 I Read/Write: R/W I Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	SDMMCWA_WCOAL_TMVAL

CHAPTER 33: SATA CONTROLLER

33.1 Overview

The Serial Advance Technology Attachment (SATA) controller enables a control path from a Tegra® X1 device to an external SATA device. An SSD/HDD/ODD/e-SATA drive can be connected. This is a one port configuration.

The SATA controller is compliant with SATA specification rev. 3.1 and AHCI specification rev. 1.3.1. This includes all errata, ENC, and TP, except DHU (direct head unload).

33.2 Device ID

The device ID for the SATA controller in Tegra X1 is 0x10E7. The AHCI driver needs to make sure the class code is set to 0x0106 and the program interface code is set to 0x01 (after boot) before it starts operating the controller in AHCI mode.

33.3 AUX Version of Pad Idle Detection

In Tegra X1, the pad supports VAUX bus idle detection. The Miscellaneous Register unit has been modified to support both the VCORE and VAUX versions of bus idle detection for detecting the OOB signaling, while adding the control to put the VCORE portion and/or the VAUX portion of the PAD in IDDQ mode when the SATA controller is power gated under the following conditions:

Table 211: SATA Controller Power-Gating Conditions

Link State	VCORE	VAUX	Note
Partial	ON	ON	VCORE portion is powered to maintain the common mode voltage. OOB detection using VCORE version with VAUX OOB detection disabled.
Slumber	OFF	ON	VAUX portion is used for OOB detection.

During LP0, the MUX between XUSB and SATA would keep the VAUX portion of the PAD in IDDQ, while the VCORE portion enters IDDQ when the system VCORE power rail is removed.

33.3.1 RX Lane Calibration

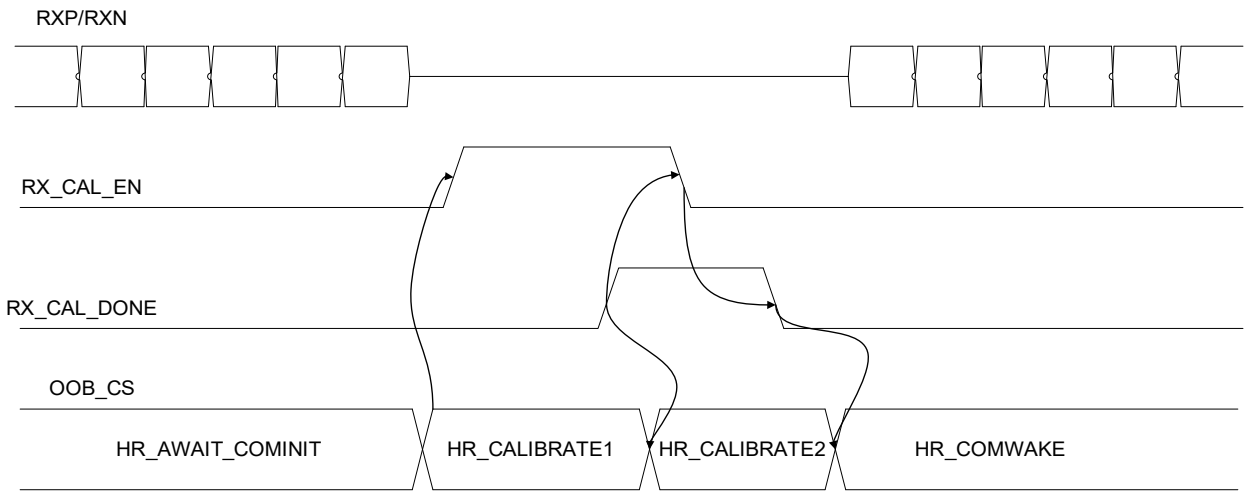
The RX lane calibration is added to the 20nm SATA pad to compensate the datapath offsets for 20nm pad. RX lane calibration should be performed during 2 instances

- Initial Power up sequence
- LP0 exit

The SATA specification supports calibration during OOB signaling. Calibration is performed after receiving the COMINIT from the drive and before sending the COMWAKE.

Also OOB signaling is performed for the above mentioned two events. So in order to support the RX lane calibration for the 20nm pad, a new calibration state has been added to the SATA OOB sequence. The SATA OOB RX calibration timing diagram is given below.

Figure 155: SATA OOB RX Calibration Timing Diagram



A Config bit enables or disables the RX calibration during OOB sequence.

33.3.2 PLL Calibration Sequence and Sleep State Addition

The 20 nm UPHY pad changes PLL power up sequences as below:

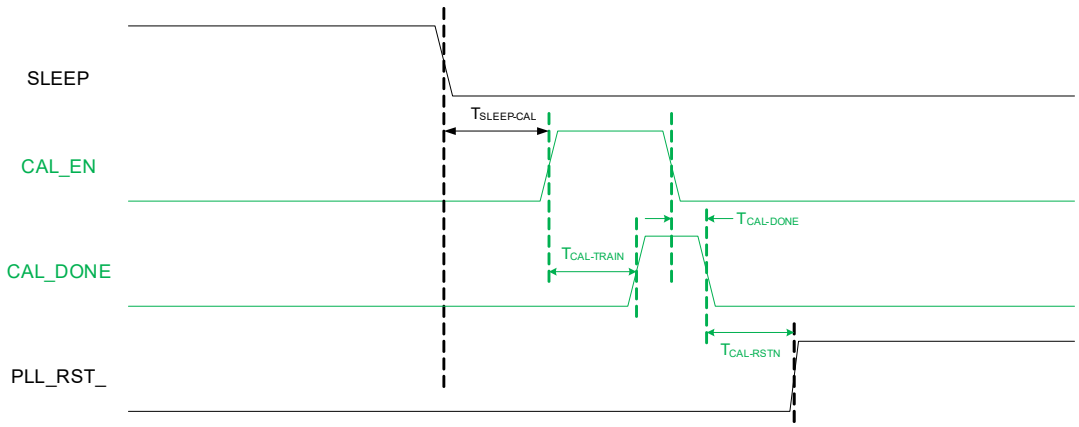
- PLL Calibration Sequence
- PLL Sleep state addition.

As with lane calibration, a PLL calibration sequence is recommended to be done during:

- Power up sequence
- LP0 exit

The following timing diagram explains the PLL Calibration training sequence:

Figure 156: PLL CAL Training Sequence



The PLL contains a low power state “SLEEP”, which drives the “PLL_SLEEP” pins to a particular value so that a particular portion of PLL can be shut off. The following table shows the different PLL_SLEEP states and their functionalities.

Table 212: PLL_SLEEP States

	Main Controls			References		Calibration		Clocks		
State	IDDQ	SLEEP	RST_	Bandgap	Reg	Resistor	PLL	VCO	Digital	Serial
Off	1	X (11)	X (0)	Off	Off	Off	Off	Off	Off	Off

Table 212: PLL_SLEEP States

	Main Controls			References		Calibration		Clocks		
Sleep3	0	11	X (0)	Off	Off	On/Off	Off	Off	Off	Off
Sleep2	0	10	X (0)	On	Off	On/Off	Off	Off	Off	Off
Sleep1	0	01	X (0)	On	On	On/Off	On/Off	Off	Off	Off
Reset	0	00	0	On	On	On/Off	On/Off	Active	Off	Off
Active	0	00	1	On	On	On/Off	On/Off	Active	Active	Auto-Control

The values driven on “PLL_SLEEP” state may be different for different IP’s. For SATA it is suggested to drive “Sleep3” value on “PLL_SLEEP” pins during SLEEP state.

33.4 Boot through SATA

When the SATA device is used for booting, the controller will operate in legacy PIO mode in the beginning phase of the booting process. Controller default class code is set to 0x0101 and the program interface code is set to 0x85 to indicate legacy PIO mode support. Boot code will need to change class code to 0x0106 and program interface code to 0x01 before it starts to operate controller in AHCI mode during the later phases of booting process. Also the AHCI driver needs to make sure 0x0106 and 0x01 are set before operating controller in AHCI mode.

33.5 MCCIF with Dual Channel Support

The Tegra X1 MCCIF has dual channel support. In order to support the 32-bit dual channel LPDDR-4 interface, the MCCIF has a dedicated response bus for each channel on the MSC bus. For a given request, the response can come from only one channel. However, multiple requests from an MCCIF can be outstanding to each channel at any point. So the MCCIF can receive responses from the two channels simultaneously.

33.6 Power-Gating Sequence

This section describes all the steps needed to achieve power gating.

- Wake events from the drives are supported
- Wake events can only happen in non LP0 mode
- No wake support in LP0 state
- The entire partition including the FPCI wrapper is power gated
- If there can be no wake event, the pads can be put in IDDQ mode. This is decided on the type of drive connected
- The wakeup of the partition (SAX) that hosts the SATA controller and the IPFS shall be initiated by software

33.6.1 Power-Gating Overview

Power gating and ungating can be initiated by software. Power gating should be done when the SATA link is in slumber mode, with no outstanding commands. The software should keep a backup of a set of registers in system memory as they lose value once the power is removed. This involves all the configuration space, extended configuration space registers and the MMIO space registers. Since the software is expected to know the state of registers inside the controller, it need not read the registers every time a power gating sequence is performed.

If hardware wake up is to be supported (for example, hot plug detection), the pads should be left in slumber. If hardware wake up is not needed, then the pads should be placed in the IDDQ mode. This is the preferred mode of operation. The bits pad_iddq_override_en and pad_iddq_override_value could be used to control the state of the pads.

33.6.2 Hardware Initiated Wake up

For a hardware-initiated wake up, the pads detect activity on the SATA link. Once the OOB sequence is detected, the bit oob_wake_detected unit is written. Note that the controller itself is not involved in this signal as it is power gated.

The signal from the pads that is of concern here is the rx_stat_idle. This signal must be monitored for any activity. This is an active low signal. There should be two bits that the software needs to read for the interrupt.

1. To log the status and send an interrupt to software (software controlled PG) or signal to the PMC (hardware controlled PG). There are three bits associated with this.
    a. A status bit SATA_L0_RX_STAT_IDLE in the register SATA_AUX_RX_STAT_INT_0. This bit shows the current status of rx_stat_idle. It is set to 1 whenever there is activity on the RX path, else it is set to 0.
    b. A sticky interrupt status bit SATA_RX_STAT_INT_STATUS in the register SATA_AUX_RX_STAT_INT_0. This is set whenever rx_stat_idle is set to 1 and will stay 1 until the interrupt is cleared by software.
    c. There are bits to set/clear the bits above. Writing a 1 to SATA_RX_STAT_INT_SET bit in SATA_AUX_RX_STAT_SET_0 register will set the status. Writing a 1 to SATA_RX_STAT_INT_CLR will clear this status.
2. To have a mask to enable / disable interrupts. This is done via central interrupt controller with the interrupt SATA_RX_STAT in PRI_ICTLR_* registers. Follow the same guidelines as any other units to enable/disable the SATA_RX_STAT interrupt.

The software clears the interrupt and the mask during normal operation.

33.6.3 Bits Required in Power Gating Sequences

The following bits are required in power gating sequences. Some of them are needed in both hardware and software sequencing methods. Some of them are required only in one of the modes of power gating. The bits could be named differently. The bits are –

- Oob_wake_en – this bit is to be placed in a non-power gated region. Its role is to enable wake events from the drive. This is enabled by writing a 1 to SATA_RX_STAT in PRI_ICTLR_* registers
- Oob_wake_detected – this bit shall be written when “oob_wake_en” is asserted and a wake is detected on the SATA link. The “rx_stat_idle” toggling shall assert this bit. In software mode of power gating, an interrupt needs to be sent to the SATA driver. This is the SATA_RX_STAT_INT_STATUS bit in register SATA_AUX_RX_STAT_INT_0.
- Pmc2sata_pg_info – this bit shall drive the signal pmc2sata_pg_info.

33.6.4 Software Initiated Power Gating / Ungating Sequence

Table 213: Steps involved in Power Gating

#	Description	Register Programming	Comments
1.	Driver decides to enter power gating, based on the fact that links are in slumber state and no commands are outstanding.		
2.	Driver programs a register in the PMC to indicate to SATA that it is entering power gating. This shall drive the pmc2sata_pg_info signal.	APBDEV_PMC_SATA_PWRGT_0.PG_INFO = 1	If driver detects a hardware wake from the drive before setting PG_INFO bit, it should bail out and should not begin the PG sequence.
3.	Driver would read out the registers in the SATA controller and save it in system memory.	Register list is in appendix.	The time taken for this would depend on the time taken for the software to read a register. Ideally, this should be done only once.
4.	Driver needs to read and set the rx_idle_t registers in the APBMISC block so idle threshold can be continuously driven while SAX is power-gated	Read SATA_AUX_MISC_CNTL_1_0 register L0_RX_IDLE_T_SAX field and write that value into same register L0_RX_IDLE_T_NPG field and write ‘1’ to L0_RX_IDLE_T_MUX field.	

Table 213: Steps involved in Power Gating

#	Description	Register Programming	Comments
5.	Driver should read the error and interrupt registers before entering power gating.	Read PxSERR and PxIS, and IS registers. Also read PxCI register to make sure that there are no outstanding commands. If (PxCI !=0) or any unexpected error, driver should bail out of PG sequence by setting APBDEV_PMC_SATA_PWRGT_0.PG_INFO = 0 Sets the bit rx_stat_idle_bypass to enable detection of a comwake.	If there is something unexpected or an error is detected or if HW wake is detected from the drive, then the software should bail out of PG sequence (clear PMC bit) Once PG_INFO=1 and/or rx_stat_idle bypass is enabled, HW wake could be detected via rx_stat_idle interrupt only. In that case, driver needs to wait until HBA is in active state (PxSSTS.DET=0x3 and PxSSTS.IPM=0x1) before issuing any new commands.
6.	The driver should gate SATA clocks and assert reset to SATA.	CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0.SATA_CLKEN= '0' CLK_RST_CONTROLLER_RST_DEVICES_V_0.SATA_RESET=1 CLK_RST_CONTROLLER_RST_DEVICES_W_0.SATACOLD_RST=1	
7.	If HW wake up (example hot-plug or async notification) is needed: The driver shall write a bit in the Interrupt Controller in non-power gated region to enable the rx_stat interrupt generation on detection of rx_stat (OOB) wake.	Clear sata_rx_stat interrupt status and enable sata_rx_stat interrupt to start detecting rx_stat_idle changes from SATA pad when detecting OOB wake sequence: Write apb_misc sata_aux register RX_STAT_CLR bit[0] "SATA_RX_STAT_INT_CLR" field to '1' Write interrupt controller register PRI_ICTLR_CPU_IER_SET_0 bit[13] "SATA_RX_STAT" field to '1' The mapped status bit for sata_rx_stat interrupt in interrupt controller is at register PRI_ICTLR_ISR_0 bit[13] "SATA_RX_STAT"	At this point, sideband (PHY logic + interrupt controller) takes over interrupt generation.
8.	If HW wake up (example hot-plug or async notification) is needed: The driver shall place the SATA PADPLL in reset. If Hw wake up is not needed: Driver shall place the SATA PHY and SATA PADPLL in IDDQ.	a) SATA_PADPLL_RESET_SWCTL =1 SATA_PADPLL_RESET_OVERRIDE_VALUE=1 b) SATA_PADPLL_RESET_SWCTL =1 SATA_PADPLL_RESET_OVERRIDE_VALUE=1 SATA_PADPHY_IDDQ_SWCTL=1 SATA_PADPHY_IDDQ_OVERRIDE_VALUE=1 Wait for time specified in SATA_LANE_IDDQ2_PADPLL_IDDQ SATA_PADPLL_IDDQ_SWCTL=1 SATA_PADPLL_IDDQ_OVERRIDE_VALUE=1	The driver controls SATA PLL at run time (no HW low power sequencing).
9.	The driver reports that SATA is being powered down. If PCIE is not being used, the driver can set the PLLE to either reset or IDDQ. If HW wake up is needed and exit latency cannot be met with PLLE IDDQ mode, then PLLE can be set to reset. If HW wake up is not needed or PLLE IDDQ mode exit latency is sufficient: PLLE can be set to IDDQ.	a) PLLE_ENABLE_SWCTL=1 CLK_RST_CONTROLLER_PLLE_BASE_0.PLLE_ENABLE=0 b) PLLE_ENABLE_SWCTL=1 CLK_RST_CONTROLLER_PLLE_BASE_0.PLLE_ENABLE=0 PLLE_IDDQ_SWCTL=1 PLLE_IDDQ_OVERRIDE_VALUE=1	Wait Time= 1 µs
10.	The driver now shall power gate the controller. This shall also involve the setting of the clamps, and assertion of the resets.	APBDEV_PMC_PWRGATE_TOGGLE_0.SAX=1 << reset overrides exist in the CAR block.	The clamp value of the hardware is such that it shall place the SATA PAD PLL in IDDQ mode but it is overridden by software value as swctl on iddq/reset is enabled above.

Table 213: Steps involved in Power Gating

#	Description	Register Programming	Comments
11.	The driver indicates to the AHCI software, that power gating is complete.		This is more of an implicit step; returning of the PG API function indicates that power-gating is done.

Table 214: Steps involved in Power Ungating

#	Description	Register Programming	Comments
1	If hardware wake up is allowed and an OOB sequence is detected, an rx_stat interrupt will be sent by the interrupt controller Or, driver decides to wake up the controller.	When sata_rx_stat interrupt is detected and serviced, interrupt routine should disable this interrupt and clear the status by doing: Write interrupt controller register PRI_ICTLR_CPU_IER_CLR_0 bit[13] "SATA_RX_STAT" field to '1' Write apb_misc sata_aux register RX_STAT_CLR bit[0] "SATA_RX_STAT_INT_CLR" field to '1'	
2	If SATA PHY and SATA PADPLL are in IDDQ mode, driver brings them out of IDDQ mode.	Get SATA PHY and PLL out of IDDQ as below: SATA_PADPLL_IDDQ_OVERRIDE_VALUE=0 Wait for programmable delay of SATA_PADPLL_IDDQ2LANE_SLUMBER_DLY. Suggested value of this delay is 3 microseconds. SATAPAD_IDDQ_OVERRIDE_VALUE=0	

Table 214: Steps involved in Power Ungating

#	Description	Register Programming	Comments
3	The driver checks if PLLE is already turned on or not. Part (a): If PLLE is already turned on, it waits for 1 ms before moving on to next step. Part (b): If PLLE is off, it brings it up.	If PCIE is running, PLLE could be already running. Check that it is running by checking its PLLE_ENABLE and PLLE_LOCK bits. Part (a): If both bits are set, then the PLLE is already turned on. Wait for 1 ms . Part (b) If neither bit is set, then do PLLE training and init sequence. If using spread spectrum, set the following bits: CLK_RST_CONTROLLER_PLLE_SS_CNTL_0. PLLE_SSCBYP=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0. PLLE_BYPASS_SS=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0. PLLE_INTERP_RESET=1 Set PLLE_LOCK_ENABLE bit in CLK_RST_CONTROLLER_PLLE_MISC register to DISABLE (0). Set the PLLE_ENABLE and PLLE_ENABLE_CML bits to DISABLE (0) in CLK_RST_CONTROLLER_PLLE_BASE register. Also set PLLE_SETUP=0 and PLLE_EXT_SETUP_17_16=0 in CLK_RST_CONTROLLER_PLLE_MISC register. Check that PLLE_IDDQ_SWCTL=1 and PLLE_IDDQ_OVERRIDE_VALUE=1. If they are not set to 1, set them to 1. Then set PLLE_IDDQ_OVERRIDE_VALUE=0 to begin the training sequence. Wait for PLLE training sequence to finish by waiting until PLLE_PLL_READY bit is set to 1 in CLK_RST_CONTROLLER_PLLE_MISC register program PLLE parameters (most are the default Reset values) – set the following values: CLK_RST_CONTROLLER_PLLE_BASE register: PLLE_PLDIV_CML = 0xd PLLE_PLDIV = 0x18 PLLE_MDIV = 0x1 PLLE_NDIV = 0xc8 CLK_RST_CONTROLLER_PLLE_MISC register PLLE_SETUP = 0x7 (This is not the default) PLLE_LOCK_ENABLE = ENABLE (1) (This isn't default) If using spread spectrum, set the following bits: CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCBYP=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_BYPASS_SS=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_INTERP_RESET=1 Enable the PLLE now by setting PLLE_ENABLE and PLLE_ENABLE_CML to ENABLE (1) in CLK_RST_CONTROLLER_PLLE_BASE register. Wait for PLLE to lock by waiting until PLLE_LOCK bit is set to 1 in CLK_RST_CONTROLLER_PLLE_MISC register. After PLLE_LOCK is detected, wait for an additional programmable delay of PLLE_LOCK_DLY. This will help in case the PLLE_LOCK gets asserted earlier. The suggested value of this additional delay is 25 microseconds. Do the following steps if you intend to use spread spectrum: program spread spectrum coefficients- CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCMAX = 0x25, CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCINCINTRV = 0x20, CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCINC = 0x1. CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_BYPASS_SS=0 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCBYP=0 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_INTERP_RESET=0 Enable CML clock for SATA by setting PLLE_CML1_OEN bit to 1 in CLK_RST_CONTROLLER_PLLE_AUX_0 register. If PLLE is already turned on, check that the SATA CML clock output is turned on. Otherwise, turn it on by setting PLLE_CML1_OEN bit to 1 in CLK_RST_CONTROLLER_PLLE_AUX_0 register.	

Table 214: Steps involved in Power Ungating

#	Description	Register Programming	Comments
4	The driver deasserts reset to SATA PADPLL and waits until it locks.	SATA_PADPLL_RESET_OVERRIDE_VALUE=0 Wait for Satapll_lockdet ='1'	Expected maximum time for SATA PADPLL to lock = 15 microseconds.
5	The driver toggles the power-gate SAX bit in the PMC to power-ungate SAX partition.	Toggle the SAX partition register bits in the PMC APBDEV_PMC_PWRGATE_TOGGLE_0.SAX=0 APBDEV_PMC_REMOVE_CLAMPING_CMD_0.SAX=1	
6	The power ungating is complete.		This is an implicit step. No action here.
7	Driver turns on the clocks to SATA and deasserts resets.	Software to ungate txclk and all SATA clocks, reset CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0.SATA_CLKEN= '1', CLK_RST_CONTROLLER_RST_DEVICES_W_0.SATACOLD_RST= '0'? CLK_RST_CONTROLLER_RST_DEVICES_V_0.SATA_RESET =0	
8	Driver restores the registers of the controller.	Driver should restore the registers in the following order: IPFS, CFG, Ext CFG, BAR5. Make sure that any registers that change values based on customer design are restored to their proper values. During the restoration of the registers, the driver would now need to restore the register T_SATA0_CFG_POWER_GATE_SSTS_ RESTORED after the ssts_det, ssts_spd are restored. This register is used to tell the controller whether a drive existed earlier or not and move the PHY state machines into either HR_slumber or not.	Expected to take ~10 µs.
9	Driver needs to switch the rx_idle_t driven source back to from Sata controller after SAX is power-ungated	Write SATA_AUX_MISC_CNTL_1_0 register L0_RX_IDLE_T_MUX field to '0'.	
10	Driver can start to use main SATA interrupt instead of the rx_stat_t interrupt.	After SATA logic power-ungated we can start to use main interrupt from SATA controller and interrupt status mapped to interrupt controller register PRI_ICTLR_ISR_0 bit[23] "SATA_CTL"	
11	Set the bits in the CAR to allow hardware based low power sequencing.	SATAPAD_PLL_PLLE_IDDQ_RESET_SWCTL =0	
12	Driver indicates to the controller that the power ungating process is complete.	Clear bit in PMC APBDEV_PMC_SATA_PWRGT_0.Pmc2sata_pg_info = 0 Clears the bit rx_stat_idle_bypass to enable detection of a comwake.	

Note: *Wake up behavior when CPU is off: Wake up might include waking the CPU itself. If the CPU is powered down, the rx_stat interrupt causes flow controller/PMC to power up CPU, the CPU goes to reset vector handler code in OS kernel patch which will restore the CPU state, reinitialize the CPU, un-powergate SATA, restore the SATA state, and give control to AHCI software. If the CPU is powered up, the rx_stat interrupt signal generates a CPU interrupt, interrupt handler in driver shall un-powergate SATA and restore SATA state.*

33.6.5 Software Changes

In software initiated power gating and ungating, software controls the entry and exit of the power gating controller. In the hardware initiated wake up mode, the software needs to read the bit oob_wake_detected to determine if the controller needs to be power ungated. The software needs to keep a backup of all the registers in system memory and needs to reprogram them once in the power ungating process. The software needs to program the registers in the correct sequence as mentioned above.

The PxCMD.CR bit does not have the right value after register restoration following power-ungating. The software should not read that bit.

A software workaround is needed when software initiating power-gating. To prevent accidental COMRESET from being sent on the SATA link, software needs to set an override for hardware not to look at SATA PLL lockdet when shutting down the SATA

PLL. This can be done by writing SATA ext. config space register T_SATA<0>_CFG_MISC (0x550): write bit [10] to '1' and bit [8] to '0' (field T_SATA<0>_CFG__PHY_RESET_USAGE_MODE).

33.7 Address Translation

The BAR spaces and PCI Config space from the PCI bus environment are mapped as-is to an address space carved out of ARM (ordering of the registers is not changed). Tegra X1 devices do not follow the PCI specifications. They do not have any concept of the PCI configuration space, BAR space, and the extended configuration space. The system addressing is of 32 bits, unlike 40 bits in the PCI bus environment. The address can map up to 4 GB.

There is an address translation mechanism in the IPFS. With this address translation, a certain address range is mapped to the configuration space registers, and likewise for the BAR 5 and the extended configuration space registers.

The address mapping that is being done is explained in the "Address Translation Table" subsection. Even though, BAR 0 to Bar 4 are not applicable to Tegra X1 devices, they too have been allocated address ranges.

33.7.1 Address Translation Table

The next table shows how all of the SATA register spaces and SATA-IPFS registers are mapped to the Tegra X1 memory map. In Tegra X1 devices, 64KB APB space is allocated for all these registers and 512B APB space is allocated for SATA IOBIST registers.

Table 215: SATA Address Translation Table

Register Section		DFPCI address [39:0]	APB adr[31:0] for DFPCI mapping	PRI address [31:0]	APB adr[31:0] for PRI mapping
			*APB address 0x7002_0000 - 0x7002_0FFF is reserved for IPFS local registers		
1. SATA Configuration Space		Address starting with 0xFD_FE or 0xFE_0			
SATA configuration space 0x0 - 0x255	Start address	0xFD_FE00_5000	0x7002_1000	0x5010_0000	0x7002_9000
	End address	0xFD_FE00_50FF	0x7002_10FF	0x5010_00FF	0x7002_90FF
SATA extended configuration space 0x100 - 0xFFF	Start address	0xFE_0100_5000	0x7002_1100	0x5010_0100	0x7002_9100
	End address	0xFE_0F00_50FF	0x7002_1FFF	0x5010_0FFF	0x7002_9FFF
		Format : 0xFE_0X00_50YZ			
		XYZ from 0x100 to 0xFFF			
I/O space		Address starting with 0xFD_FC			
2. SATA BAR0					
Register MCP_SATA_PRI_ COMMAND_0	Address in SATA Compatible mode	0xFD_FC00_01F0	0x7002_21F0	0x5020_0200	0x7002_A200
	Address in SATA Native mode	SATA BAR0 with offset 0x0	0x7002_2000	0x5000_0200	0x7002_A200
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_2000	Not used but address can be same as native mode	0x7000_A200

Table 215: SATA Address Translation Table

Register Section		DFPCI address [39:0]	APB adr[31:0] for DFPCI mapping	PRI address [31:0]	APB adr[31:0] for PRI mapping
Register MCP_SATA_PRI_ COMMAND_1	Address in SATA Compatible mode	0xFD_FC00_01F4	0x7002_21F4	0x5020_0204	0x7002_A204
	Address in SATA Native mode	SATA BAR0 with offset 0x4	0x7002_2004	0x5000_0204	0x7002_A204
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_2004	Not used but address can be same as native mode	0x7000_A204
3. SATA BAR1					
Register MCP_SATA_PRI_ CONTROL	Address in SATA Compatible mode	0xFD_FC00_03F4	0x7002_33F4	0x5020_0300	0x7002_A300
	Address in SATA Native mode	SATA BAR1 with offset 0x0	0x7002_3000	0x5020_0300	0x7002_A300
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_3000	Not used but address can be same as native mode	0x7002_A300
4. SATA BAR2					
Register MCP_SATA_SEC_ COMMAND_0	Address in SATA Compatible mode	0xFD_FC00_0170	0x7002_4170	0x5020_0400	0x7002_A400
	Address in SATA Native mode	SATA BAR2 with offset 0x0	0x7002_4000	0x5020_0400	0x7002_A400
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_4000	Not used but address can be same as native mode	0x7002_A400
Register MCP_SATA_SEC_ COMMAND_1	Address in SATA Compatible mode	0xFD_FC00_0174	0x7002_4174	0x5020_0404	0x7002_A404
	Address in SATA Native mode	SATA BAR2 with offset 0x4	0x7002_4004	0x5020_0404	0x7002_A404
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_4004	Not used but address can be same as native mode	0x7002_A404
5. SATA BAR3					
Register MCP_SATA_SEC_ CONTROL	Address in SATA Compatible mode	0xFD_FC00_0374	0x7002_5374	0x5020_0500	0x7002_A500
	Address in SATA Native mode	SATA BAR3 with offset 0x0	0x7002_5000	0x5020_0500	0x7002_A500
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_5000	Not used but address can be same as native mode	0x7002_A500
6. SATA BAR4					
Bus Master Registers 0x0 - 0xF	Address in SATA Compatible mode	SATA BAR4 with offset 0x0 - 0xF	0x7002_6000 - 0x7002_600F	0x5020_0600 - 0x5020_060F	0x7002_A600 - 0x7002_A60F
	Address in SATA Native mode	SATA BAR4 with offset 0x0 - 0xF	0x7002_6000 - 0x7002_600F	0x5020_0600 - 0x5020_060F	0x7002_A600 - 0x7002_A60F
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_6000 - 0x7002_600F	Not used but address can be same as native mode	0x7002_A600 - 0x7002_A60F

Table 215: SATA Address Translation Table

Register Section		DFPCI address [39:0]	APB adr[31:0] for DFPCI mapping	PRI address [31:0]	APB adr[31:0] for PRI mapping
Memory space		Address[39:32] = 0x0			
7. SATA BAR5 (claim 8KB)					
AHCI Registers 0x0 - 0x1FFF	Start address	SATA BAR5 with offset 0x0	0x7002_7000	0x5030_0000	0x7002_B000
	End address	SATA BAR5 with offset 0x1FFF	0x7002_8FFF	0x5030_1FFF	0x7002_CFFF
8. SATA IOBIST registers	Start address	Non accessible	N.A.	0x5050_0000	0x7002_D000
	End address	Non accessible	N.A.	0x5050_01F7	0x7002_D1F7
Note: In Tegra X1 devices, there is a standalone APB client created for SATA IOBIST register access, the PRI interface mapping from the SATA APB client is no longer valid					
The SATA IOBIST registers are mapped to 0x7000_6C00 - 0x7000_6DFF					

33.8 Programming Guidelines

33.8.1 Cold Boot

The IOPHY/UPHY of SATA is shared with XUSB. Refer to Chapter 22: USB Complex in this TRM for the IOPHY/UPHY PLL initialization sequence.

The IOPHY/UPHY lane ownership assignment is located in XUSB PADCTL, where the PAD parameter programming registers have also been moved to XUSB PADCTL. Refer to Chapter 22: USB Complex in this TRM for the IOPHY/UPHY PAD initialization sequence.

The PAD driver assigns the SPIO pins to the SATA controller:

- DA uses an available GPIO pin, if required:

The PAD driver assigns the PHY to the SATA controller:

- Program the following XUSB PADCTL registers to assign the static UPHY PAD and PLL parameters according to the platform-specific configuration (note the following lists all PAD and PLL registers; the driver should only program the ones whose value are different from the default values):
  - XUSB_PADCTL_UPHY_PLL_S0_CTL1_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL2_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL3_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL4_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL5_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL6_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL7_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL8_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL9_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL10_0
  - XUSB_PADCTL_UPHY_PLL_S0_CTL11_0
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_1_0
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_3_0
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_4_0

- XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_5_0
- XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_6_0
- XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_7_0
- XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_8_0
- XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_9_0

- Program the following XUSB PADCTL register to assign the PHY to SATA:
  - XUSB_PADCTL_USB3_PAD_MUX_0[SATA_PAD_LANE0] to 'SATA'

The PAD driver programs the clocks and deasserts the resets to the controllers:

- Set the following CAR register bits to '1' to enable the clocks to *SATA:*
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA]
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA_OOB]
- Program the following CAR register bits to set the source of the *SATA* clocks, where PLLP_OUT0 runs at 408 MHz:
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0[SATA_CLK_SRC] to 'PLLP_OUT0'
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0[SATA_CLK_DIVISOR] to '0x6'
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0[SATA_OOB_CLK_SRC] to 'PLLP_OUT0'
  - CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0[SATA_OOB_CLK_DIVISOR] to '0x2'
- Set the following CAR register bits to '0' to deassert reset to *SATA:*
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_OOB_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_SATACOLD_RST]

The PAD driver brings the UPHY out of IDDQ:

- Program the following XUSB PADCTL registers to '1' to bring SATA IOPHY out of IDDQ.
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]
  - Wait 200 µs

The SATA PEP driver initializes the IPFS registers:

- Program the following SATA IPFS registers to allow software accesses to the SATA's MMIO registers:
  - SATA_AXI_BAR5_START_0[AXI_BAR5_START] to '0x70020'
  - SATA_AXI_BAR5_SZ_0[AXI_BAR5_SIZE] to '0x00008'
  - SATA_FPCI_BAR5_0[FPCI_BAR5_START] to '0x40020000'
  - SATA_FPCI_BAR5_0[FPCI_BAR5_ACCESS_TYPE] to '0'
- Program the following SATA IPFS register to enable the SATA:
  - SATA_CONFIGURATION_0[EN_FPCI] to '1'

The SATA PEP driver initializes SATA's configuration registers.

- Program the following SATA configuration registers to initialize SATA:
  - T_SATA<0>_CFG_1_BUS_MASTER to '1'
  - T_SATA<0>_CFG_1_MEMORY_SPACE to '1'
  - T_SATA<0>_CFG_9_BASE_ADDRESS to '0x40020000'

The SATA PEP driver initializes SATA's AUX registers.

- Program the following SATA_AUX register to disable RX idle detection interrupt:
  - SATA_AUX_RX_STAT_INT_0[SATA_RX_STAT_INT_DISABLE] to ‘1’

The SATA PEP driver enables SATA’s second level clock gating.

- Program the following IPFS registers for IPFS clock gating:
  - SATA_CONFIGURATION_0[CLKEN_OVERRIDE] to ‘1’
- Program the following AHCI registers for TX clock gating:
  - T_AHCI_HBA_PLL_CTRL_SHUTDOWN_TXCLK_ON_SLUMBER to ‘1’
  - T_AHCI_HBA_PLL_CTRL_NO_CLAMP_SHUTDOWN to ‘1’

The SATA PEP driver programs SATA’s configuration registers for WARs.

- Program the following SATA_configuration registers to fix SQUELCH Filter
  - T_SATA0_CFG_PHY_0_MASK_SQUELCH to ‘1’
  - T_SATA<0>_NVOOB_COMMA_CNT to ‘3’b100’
  - T_SATA<0>_NVOOB_SQUELCH_FILTER_LENGTH to ‘2’b11’
  - T_SATA<0>_NVOOB_SQUELCH_FILTER_MODE to ‘2’b01’
- Program the following SATA_configuration registers to fix Gen3 devices detected as Gen1 devices
  - T_SATA0_CFG_PHY_0_USE_7BIT_ALIGN_DET_FOR_SPD to ‘1’
- Program the following SATA AHCI registers to disable SATA declaring support of DEVSLP
  - T_SATA0_AHCI_HBA_CAP2_BKDR_SUPPORTS_DEVSLP to ‘0’

33.8.2 Boot ROM Boot

The Boot ROM checks if the boot setting is set to boot from SATA.

- Read the following MISC register to determine if there is a boot from SATA:
  - APB_MISC_PP_STRAPPING_OPT_A_0[BOOT_SELECT] is ‘SATA’

The Boot ROM enables the memory controllers’ AHB redirection.

- Program the following CAR register to override MC ARC clock gating
  - CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRD_0[ARC_CLK_OVR_ON] to ‘1’.
- Program the following MC registers to enable the ARC path:
  - MC_IRAM_BOM_0[IRAM_BOM] to ‘0x4000_0’.
  - MC_IRAM_TOM_0[IRAM_TOM] to ‘0x4003_F’.

The Boot ROM initializes SATA by following steps listed in Section 33.8.1: Cold Boot above.

33.8.3 ELPG Entry

The SATA PEP driver performs context saves for SATA registers specified in Section 33.9.1: Registers for Save and Restore.

The SATA PEP driver switches PAD control to SATA’s AUX registers:

- Program the following SATA_AUX register to enable RX idle detection interrupt:
  - SATA_AUX_RX_STAT_INT_0[SATA_RX_STAT_INT_DISABLE] to ‘1’
- Program the following XUSB PADCTL registers to put the Vcore side of the UPHY to IDDQ:
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ_OVRD] to ‘1’
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ] to ‘1’

- XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ_OVRD] to ‘1’
- XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ] to ‘1’

The System Power Management driver flushes MCCIF and partition clients.

- Set the following MC register bits to ‘1’ to enable flush to SATA
  - MC_CLIENT_HOTRESET_CTRL_0[SATA_FLUSH_ENABLE]
- Read the following MC register to be ‘1’ to ensure flush to SATA is enabled
  - MC_CLIENT_HOTRESET_STATUS_0[SATA_HOTRESET_STATUS]

The System Power Management driver asserts reset to SATA then disables its clocks:

- Set the following CAR register bits to ‘1’ to assert reset to SATA:
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_OOB_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_SATACOLD_RST]
- Set the following CAR register bits to ‘1’ to disable the clocks to individual SATA partitions:
  - CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0[CLR_CLK_ENB_SATA]
  - CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0[CLR_CLK_ENB_SATA_OOB]

The System Power Management driver uses software overrides to power down the IOPHY/UPHY PLL.

- Set the following CAR register bits to ‘1’ to power down the PLL:
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_PADPLL_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_LANE_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_RESET_PD_INPUT_VALUE]

The System Power Management driver disables the SATA power rails:

- Program the following PMC register bits in a single write to disable the power rail to SATA:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘SAX’
- Read the following PMC register bits to confirm the power gating status of SATA:
  - APBDEV_PMC_PWRGATE_STATUS_0[SAX] equals ‘OFF’

33.8.4 ELPG Exit

The System Power Management driver uses software overrides to enable the IOPHY/UPHY PLL.

- Set the following *CAR* register bits to ‘0’ to enable the PLL
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_PADPLL_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_LANE_PD_INPUT_VALUE]
  - CLK_RST_CONTROLLER_SATA_PLL_CFG0_0[SATA_SEQ_RESET_PD_INPUT_VALUE]

The System Power Management driver enables the SATA power rails:

- Program the following PMC register bits in a single write to disable the power rail to SATA:
  - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
  - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘SAX’
- Read the following PMC register bits to confirm the power gating status of SATA:
  - APBDEV_PMC_PWRGATE_STATUS_0[SAX] equals ‘ON’

The System Power Management driver enables the SATA clocks:

- Set the following *CAR* register bits to '1' to enable the clocks to the SATA partition:
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA]
  - CLK_RST_CONTROLLER_CLK_ENB_V_SET_0[SET_CLK_ENB_SATA_OOB]

The System Power Management driver removes power clamps to SAX partitions:

- Set the following PMC register bits to '1' to remove the power clamps to the SATA partitions:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [SAX]
- Read the following PMC register bits to confirm the power clamps to the SATA partition are removed:
  - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [SAX] equals '0'

The System Power Management driver deasserts reset to SATA.

- Set the following *CAR* register bits to '0' to assert reset to SATA:
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_V_0[SWR_SATA_OOB_RST]
  - CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_SATACOLD_RST]

The System Power Management driver disables flushes of MCCIF and partition clients.

- Set the following MC register bits to '0' to disable flush to SATA
  - MC_CLIENT_HOTRESET_CTRL_0[SATA_FLUSH_ENABLE]

The SATA PEP driver performs context restores for SATA registers specified in Section 33.9.1: Registers for Save and Restore starting from the IPFS registers followed by the SATA configuration registers and finally the SATA MMIO registers.

The SATA PEP driver switches PAD control to SATA:

- Program the following XUSB_PADCTL registers to bring the Vcore side of the UPHY out of IDDQ:
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ_OVRD] to '0'
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[RX_IDDQ] to '0'
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ_OVRD] to '0'
  - XUSB_PADCTL_UPHY_MISC_PAD_S0_CTL_2_0[TX_IDDQ] to '0'
- Program the following SATA_AUX registers to disable RX idle detection interrupt and clear RX idle interrupt:
  - SATA_AUX_RX_STAT_INT_0[SATA_RX_STAT_INT_DISABLE] to '0'
  - SATA_AUX_RX_STAT_CLR_0[SATA_RX_STAT_INT_CLR] to '1'

33.8.5 LP0

As part of the LP0 entry sequence, after setting the SATA controller to ELPG, the PAD driver should put the IOPHY to IDDQ:

- Program the following XUSB PADCTL registers to '0' to bring SATA IOPHY to IDDQ.
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]

As part of the LP0 exit sequence, after restoring power to SATA, the PAD driver brings the IOPHY out of IDDQ:

- Program the following XUSB PADCTL registers to '1' to bring SATA IOPHY out of IDDQ:
  - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_SATA_PAD_IDDQ_DISABLE_MASK0]
  - Wait 200 µs

33.8.6 Handling of Unsolicited COMINITs

To detect unsolicited COMINITs, software should check whether PxIS.PCS is set to '1' on an interrupt. The Host Bus Adaptor (HBA) cannot guarantee that a device received a COMRESET because a COMINIT may appear to be solicited to the HBA if it happens to occur closely to an issued COMRESET. Therefore, when software detects that PxIS.PCS is set, software should first issue a COMRESET to ensure that the device receives a COMRESET. Then software should perform the appropriate actions to clear PxIS.PCS to '0'. To recover, software should perform error recovery actions for a fatal error condition (including restarting the controller). Then software should perform a re-enumeration to check whether a new device has been inserted.

33.9 SATA Registers

33.9.1 Registers for Save and Restore

The registers needed to be saved and restored are:

IPFS registers

Save and restore	Offset (hex)	Save and restore	Offset (hex)
SATA_FPCI_BAR5_0	94	SATA_MSI_EN_VEC6_0	158
SATA_MSI_BAR_SZ_0	C0	SATA_MSI_EN_VEC7_0	15C
SATA_MSI_AXI_BAR_ST_0	C4	SATA_CONFIGURATION_0	180
SATA_MSI_FPCI_BAR_ST_0	C8	SATA_FPCI_ERROR_MASKS_0	184
SATA_MSI_EN_VEC0_0	140	SATA_INTR_MASK_0	188
SATA_MSI_EN_VEC1_0	144	SATA_IPFS_INTR_ENABLE_0	198
SATA_MSI_EN_VEC2_0	148	SATA_CFG_REVID_0	1A0
SATA_MSI_EN_VEC3_0	14C	SATA_CLKGATE_HYSTERSIS_0	1BC
SATA_MSI_EN_VEC4_0	150	SATA_SATA_MCCIF_FIFOCTRL_0	1DC
SATA_MSI_EN_VEC5_0	154		

Configuration Space and Extended Configuration Space Registers

Note: *Registers present in both the SATA and SATA0 address spaces are listed as SATA<0>.*

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_SATA<0>_CFG_1	4	T_SATA<0>_CFG_LINK_0	174
T_SATA<0>_CFG_3	C	T_SATA<0>_CFG_LINK_1	178
T_SATA<0>_CFG_9	24	T_SATA<0>_CFG_LINK_2	17c
T_SATA<0>_CFG_10	28	MCP_SATA<0>_CFG_TRANS_0	1D0
T_SATA<0>_CFG_12	30	T_SATA0_ALPM_CTRL	238
T_SATA<0>_CFG_13	34	T_SATA0_AHCI_HBA_CTL_0	30C
T_SATA<0>_CFG_14	38	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL	318
T_SATA<0>_CFG_15	3C	T_SATA0_AHCI_HBA_SPARE_1	320
T_SATA<0>_CFG_16	40	T_SATA0_AHCI_HBA_SPARE_2	324
T_SATA<0>_CFG_17	44	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP	328
T_SATA<0>_CFG_18	48	T_SATA0_AHCI_CFG_ERR_CTRL	32C
T_SATA<0>_MSI_CTRL	B0	T_SATA0_AHCI_HBA_CTL_1	338
T_SATA<0>_MSI_ADDR1	B4	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL	340
T_SATA<0>_MSI_ADDR2	B8	T_SATA0_CFG_FPCI_0	430

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_SATA<0>_MSI_DATA	BC	T_SATA0_CFG_ESATA_CTRL	494
T_SATA<0>_MSI_QUEUE	C0	T_SATA<0>_CTL1	4A0
T_SATA<0>_MSI_MAP	EC	T_SATA<0>_CFG_CTL_GLUE	4B0
T_SATA0_CFG_PHY_POWER	124	T_SATA<0>_PHY_CTRL	534
T_SATA0_CFG_PHY_POWER_1	128	T_SATA<0>_CTRL	540
T_SATA<0>_CFG_PHY_1	12C	T_SATA<0>_LOW_POWER_COUNT	554

Save and Restore Following Protocol

(Program T_SATA<0>_INDEX to select port 0)

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_SATA<0>_CHXCFG1	530	T_SATA<0>_CHX_PHY_CTRL_3	6B0
T_SATA<0>_CHX_MISC	684	T_SATA<0>_CHX_PHY_CTRL_4	6B4
T_SATA<0>_CHXCFG3	700	T_SATA<0>_CHX_PHY_CTRL_5	6B8
T_SATA<0>_CHXCFG4_CHX	704	T_SATA<0>_CHX_PHY_CTRL_6	6BC
T_SATA<0>_CHX_PHY_CTRL1_GEN1	690	T_SATA<0>_PRBS_CHX – OP	714
T_SATA<0>_CHX_PHY_CTRL1_GEN2	694	T_SATA<0>_CHX_LINK0	750
T_SATA<0>_CHX_PHY_CTRL1_GEN3	698	T_SATA<0>_CHX_GLUE	7F0
T_SATA<0>_CHX_PHY_CTRL_2	69C		

BAR 5 Space Registers

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_AHCI_HBA_CCC_PORTS	18	T_AHCI_PORT_PXCLBU	104
T_AHCI_HBA_GHC	4	T_AHCI_PORT_PXFB	108
T_AHCI_HBA_CCC_CTL - OP (optional)	14	T_AHCI_PORT_PXFBU	10C
T_AHCI_HBA_EM_LOC	1C	T_AHCI_PORT_PXIE	114
T_AHCI_HBA_EM_CTL - OP	20	T_AHCI_PORT_PXCMD	118
T_AHCI_PORT_PXCLB	100	T_AHCI_PORT_PXSCTL	12C

Power-Gating Registers

These registers require save and restore across the power-gating sequence. Apart from this, the software is expected to save and restore the configuration space registers.

Save and restore	Offset (hex)	Save and restore	Offset (hex)
T_AHCI_HBA_SPARE_0	a4	T_AHCI_HBA_CCC_PORTS	18
T_AHCI_HBA_GHC	4	T_AHCI_HBA_CCC_CTL - OP (optional)	14
T_AHCI_HBA_EM_LOC	1c	T_AHCI_HBA_EM_CTL - OP	20
T_AHCI_PORT_PXCLB	100	T_AHCI_PORT_PXCLBU	104
T_AHCI_PORT_PXFB	108	T_AHCI_PORT_PXFBU	10c
T_AHCI_PORT_PXIE	114	T_AHCI_PORT_PXCMD	118
T_AHCI_HBA_SHUTDOWN_TIMER	ac	T_AHCI_HBA_PLL_CTRL	a8
T_AHCI_PORT_PXSCTL	12c	T_AHCI_PORT_PXFBS	140
T_AHCI_PORT_MP	17c		

Save and Restore via Backdoor Writes

Save and restore	Offset (hex)
T_AHCI_HBA_CAP_0 (T_SATA0_AHCI_HBA_CAP_BKDR)	0 (300*)
T_AHCI_HBA_PI (T_SATA0_AHCI_HBA_PI_BKDR)	0C (33C*)
T_AHCI_HBA_CAP2 (T_SATA0_AHCI_HBA_CAP2_BKDR)	24 (330*)
T_AHCI_PORT_PXTFD (T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR)	120 (790*)
T_AHCI_PORT_PXSIG (T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR)	124 (794*)
T_AHCI_PORT_PXSSTS (T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR)	128 (798*)

Note: ** Backdoor addresses for restore*

Registers to Read Only (Not Needed to Restore)

T_AHCI_HBA_BOHC 28
T_AHCI_PORT_INTR 174
T_AHCI_PORT_PXSERR 130

33.9.2 Registers Used in Power Gating/Ungating Sequence

Refer to the PMC and Clock and Reset Controller sections for the details of these registers:

- APBDEV_PMC_SATA_PWRGT_0
- CLK_RST_CONTROLLER_SATA_PLL_CFG0_0
- CLK_RST_CONTROLLER_PLLE_AUX_0

33.9.3 SATA Register Descriptions

33.9.3.1 SATA_AXI_BARi_SZ_0 Registers

There are eight SATA_AXI_BARi_SZ_0 registers (I = 0 through 7). The size of the address range associated with BARi is in 4K increments. A value of 0 signifies BARi is not used.

SATA_AXI_BAR0_START_0 through SATA_AXI_BAR4_START_0

- For BAR0 through BAR4 (i = 0 through 4):

Offset: 0x0 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000001)

Bit	Reset	Description
19:0	0x1	AXI_BARi_SIZE

SATA_AXI_BAR5_START_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000002 (0b00000000000000000010)

Bit	Reset	Description
19:0	0x2	AXI_BAR5_SIZE

SATA_AXI_BAR6_START_0 through SATA_AXI_BAR7_START_0

- For BAR6 through BAR7 (i = 6 through 7)

Offset: 0x18 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BARi_SIZE

33.9.3.2 SATA_AXI_BARi_START_0 Registers

There are eight SATA_AXI_BARi_START registers. The AXI target address is compared to the start/size for each BAR to determine if the access is to that BAR.

SATA_AXI_BAR0_START_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x70022000 (0b01110000000000100010)

Bit	Reset	Description
31:12	0x70022	AXI_BAR0_START:The start of AXI address space for BAR0.

SATA_AXI_BAR1_START_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x70023000 (0b01110000000000100011)

Bit	Reset	Description
31:12	0x70023	AXI_BAR1_START:The start of AXI address space for BAR1.

SATA_AXI_BAR2_START_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x70024000 (0b01110000000000100100)

Bit	Reset	Description
31:12	0x70024	AXI_BAR2_START:The start of AXI address space for BAR2.

SATA_AXI_BAR3_START_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x70025000 (0b01110000000000100101)

Bit	Reset	Description
31:12	0x70025	AXI_BAR3_START:The start of AXI address space for BAR3.

SATA_AXI_BAR4_START_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x70026000 (0b01110000000000100110)

Bit	Reset	Description
31:12	0x70026	AXI_BAR4_START: The start of AXI address space for BAR4.

SATA_AXI_BAR5_START_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x70027000 (0b01110000000000100111)

Bit	Reset	Description
31:12	0x70027	AXI_BAR5_START: The start of AXI address space for BAR5.

SATA_AXI_BAR6_START_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
31:12	0x0	AXI_BAR6_START: The start of AXI address space for BAR6.

SATA_AXI_BAR7_START_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
31:12	0x0	AXI_BAR7_START: The start of AXI address space for BAR7.

33.9.3.3 SATA_FPCI_BARi_0 Registers

There are eight SATA_FPCI_BARi_0 registers. All registers have these two fields:

- Bits [31:4]: FPCI_BARi_START: This field contains the start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 12 of the value of this field.
- Bit 0: FPCI_BARi_ACCESS_TYPE: This field indicates if the address region is memory mapped versus configuration or I/ O space.
  0 = Memory-mapped access (PW only)
  1 = I/O / config access (NPW only)

SATA_FPCI_BAR0_0 through SATA_FPCI_BAR5_0

For BAR0 through BAR5 (i = 0 through 5)

Offset: 0x80 + (i * 0x4) | Read/Write: R/W | Reset: 0xfdfc0001 (0b11111101111111000000000000000xxx1)

Bit	Reset	Description
31:4	0xfdfc000	FPCI_BARi_START
0	0x1	FPCI_BARi_ACCESS_TYPE

SATA_FPCI_BAR6_0 through SATA_FPCI_BAR7_0

Offset: 0x98 + (i* 0x4) | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BARi_START
0	0x1	FPCI_BARi_ACCESS_TYPE

33.9.3.4 SATA_MSI_BAR_SZ_0

MSI BAR Size

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE: The size of the address range associated with MSI BAR is in 4K increments. A value of 0 signifies MSI BAR is not used.

33.9.3.5 SATA_MSI_AXI_BAR_ST_0

MSI AXI BAR Start

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
31:12	0x0	MSI_AXI_BAR_START: The start of upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

33.9.3.6 SATA_MSI_FPCI_BAR_ST_0

MSI FPCI BAR Start

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:4	0x0	MSI_FPCI_BAR_START: The start of upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

33.9.3.7 SATA_MSI_VECi_0 Registers

There are eight SATA_MSI_VECi_0 vector registers (i = 0 through 7). Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223. When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

Offset: 0x100 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTORi

33.9.3.8 SATA_MSI_EN_VEC0_0

There are eight SATA_MSI_EN_VECi_0 vector registers (i = 0 through 7). Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223. When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

Offset: 0x140 + (i * 0x4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTORi

33.9.3.9 SATA_CONFIGURATION_0

Configuration

Offset: 0x180 | Read/Write: R/W | Reset: 0x800X8X40 (0b1xxxxxxxxxxx1xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x1	CLKEN_OVERRIDE: This can override the clock enable in case of malfunction.
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on the AFI upstream. A value of 1b indicates there are no outstanding reads to the initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on the AFI upstream. A value of 1b indicates there are no outstanding writes to the initiator.
15	RW	0x1	WDATA_LEAD_CYA: Used to enable/disable the handling of write data ahead of requests on the IPFS AXI target.
14	RW	0x0	WR_INTRLV_CYA: Used to enable/disable the handling of interleaved write requests on the IPFS AXI target
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to the IPFS target. A value of 1b indicates there are no outstanding reads to the downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to IPFS target. A value of 1b indicates there are no outstanding writes to the downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any active bits valid or not.
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = whenever MSI is ready assert the interrupt 0 = default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: Input to upstream FPCI 1 = whenever a write is ready, send it 0 = write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to the upstream FPCI. Allow the upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for upstream FPCI. Allow upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: Used for the downstream FPCI. Allow downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: Input to the downstream FPCI. Allow downstream FPCI responses to pass writes.
1	RW	0x0	DFPCI_PASSPW: Input to the downstream FPCI. Allow downstream FPCI reads to pass writes.

Bit	R/W	Reset	Description
0	RW	0x0	EN_FPCI: When the IPFS device block is disabled, it is completely invisible on the IPFS bus; that is, it does not even process IPFS configuration accesses.

33.9.3.10 SATA_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows an FPCI error to be forwarded to the AXI response when the FPCI error response indicates Master Abort. 1 = forward error 0 = return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows an FPCI error to be forwarded to the AXI response when the FPCI error response indicates Data Error. 1 = forward error 0 = return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows an FPCI error to be forwarded to the AXI response when the FPCI error response indicates Target Abort. This bit also covers decode errors generated when there is no DEVSEL received. 1 = forward error 0 = return AXI OKAY response (2'b0)

33.9.3.11 SATA_INTR_MASK_0

Interrupt Masks

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0xxxxxxx0)

Bit	Reset	Description
16	0x0	IP_INT_MASK: IP (SATA/AZA) interrupt to the CPU gated by a mask.
8	0x0	MSI_MASK: MSI to the CPU gated by a mask.
0	0x0	INT_MASK: Interrupt to the CPU gated by a mask.

33.9.3.12 SATA_INTR_CODE_0

Interrupt Control

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0b00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: Interrupt codes. If the code is 0, logging of the next interrupt is enabled 0 = INT_CODE_CLEAR: Clear interrupt code 1 = INT_CODE_INI_SLVERR: Interrupt code for CPU AXI SLVERR response to IPFS 2 = INT_CODE_INI_DECERR: Interrupt code for CPU AXI DECERR response to IPFS 3 = INT_CODE_TGT_SLVERR: Interrupt code for PCIe endpoint FPCI target abort or data error response to IPFS 4 = INT_CODE_TGT_DECERR: Interrupt code for PCIe2 FPCI master abort response to IPFS 5 = INT_CODE_TGT_WRERR: Interrupt code for bufferable write to non-posted write address region 6 = RSVD1: Reserved 7 = INT_CODE_DFPCI_DECERR: Interrupt code for PCIe2 response to downstream request when downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR: Interrupt code for IPFS response to downstream request when AXI target address does not fall in any of IPFS downstream BARs 9 = INT_CODE_FPCI_TIMEOUT: Interrupt code for FPCI Timeout 10 = RSVD2: Reserved for future expansion 11 = RSVD3 12 = RSVD4 13 = RSVD5 14 = RSVD6 15 = INT_CODE_SM_FATAL_ERROR: Interrupt code for SM fatal error 16 = INT_CODE_SM_NON_FATAL_ERROR: Interrupt code for SM non-fatal error

33.9.3.13 SATA_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, this field contains address bits [31:2], either in FPCI memory space or AXI space. For FPCI generated errors, this field contains the FPCI address. For AXI/IPFS generated errors, this field contains the AXI address.
0	0x0	DIR: Indicates the direction of the AXI/FPCI transaction. If the signature type is 6 (sideband message), this field is 1. 1=rd 0=wr 0 = WRITE: Interrupt due to a write transaction 1 = READ: Interrupt due to a read transaction

33.9.3.14 SATA_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of the captured FPCI address (bits [39:32])when the interrupt code is 3, 4, or 7.

33.9.3.15 SATA_IPFS_INTR_ENABLE_0

IPFS Interrupt Enable

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxx00000000)

Bit	Reset	Description
13	0x0	EN_SM_NON_FATAL_ERROR: Enable bit for interrupt code 15
12	0x0	EN_SM_FATAL_ERROR: Enable bit for interrupt code 14
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

33.9.3.16 SATA_UFPCI_CONFIG_0

Upstream FPCI Configuration

Offset: 0x19c | Read/Write: R/W | Reset: 0x00000002 (0b00010)

Bit	Reset	Description
4:0	0x2	UNITID_T0C0: Upstream FPCI Unit ID for controller 0. HyperTransport, upstream FPCI request

33.9.3.17 SATA_CFG_REVID_0

CFG_REVID register

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x000X10XX (0bxxxxxx0100xxxxxxx1)

Bit	R/W	Reset	Description
19	RO	X	DEV2SM_NONISO_REQUEST_PEND: Indicates if a non ISO request is pending. 0 = NO 1 = YES
18	RO	X	DEV2SM_ISO_REQUEST_PEND: Indicates if an ISO request is pending. 0 = NO 1 = YES
13:12	RW	0x1	STRAP_CPU_MODE: MCP: Mode to send MSI. Can be programmable 0 = NB_INTEL 1 = NB_AMD 2 = AMD 3 = TMTA
11	RW	0x0	CFG_REVID_WRITE_ENABLE: MCP: Enable to override the rev ID. Can be programmable 0 = CLEAR 1 = SET
10	RW	0x0	CFG_REVID_OVERRIDE: MCP: Can override the current revision ID. Can be programmable 0 = DISABLE 1 = ENABLE
4	RO	X	DEV2LEG_NONCOH_REQUEST_PEND: MCP: Tells the leg block that a non-coherent request is pending. 0 = NO 1 = YES
3	RO	X	DEV2LEG_COH_REQUEST_PEND: MCP comment: Tells the leg block that a coherent request is pending 0 = NO 1 = YES
2	RW	0x1	SM2DEV_FPCI_TIMEOUT_EN: FPCI timeout enable bit for Controller 0 = DISABLE 1 = ENABLE

33.9.3.18 SATA_FPCI_TIMEOUT_0

FPCI_TIMEOUT register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x000f0000 (0b11110000000000000000)

Bit	Reset	Description
19:0	0xf0000	SM2ALL_FPCI_TIMEOUT_THRESH: This sets the timeout threshold value for the FPCI bus. The counter starts counting when each queue (iso/niso- rd/wr) has a pending request in the FPCI wrapper. The count resets when the requests are popped.

33.9.3.19 SATA_TOM_0

Top of Memory Limit

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x3fff0fff (0b11111111111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3fff	LEG2ALL_TOM2: Top of Memory Limit 2.
11:0	0xfff	LEG2ALL_TOM1: Top of Memory Limit 1.

33.9.3.20 SATA_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x1ac | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND: Number of pending initiator ISO PW responses

33.9.3.21 SATA_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x1b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND: Number of pending initiator NISO PW responses

33.9.3.22 SATA_INTR_STATUS_0

IPFS Interrupt Status

Offset: 0x1b4 | Read/Write: RO | Reset: 0x0000000X (0bxxx)

Bit	Reset	Description
2	X	IP_INTR_STATUS: Status of IP (SATA/AZA) interrupt
1	X	MSI_INTR_STATUS: Status of MSI interrupt
0	X	IPFS_INTR_STATUS: Status of IPFS interrupt

33.9.3.23 SATA_DFPCI_BEN_0

Downstream FPCI Byte Enables

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	0x0	EN_DFPCI_BEN: Enable bit for BEN; when set, programmed BE is sent on the DFPCI bus
3:0	0x0	DFPCI_BYTE_ENABLE_N: Active low byte enables

33.9.3.24 SATA_CLKGATE_HYSTERESIS_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000014 (0b00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of IPFS clock cycles to wait after clock gating criteria is met to disable IPFS/ FPCI clocks

33.9.3.25 SATA_SATA_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register.

Note: *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility*

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and the MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside the MCCIF and PC. A '1' written to the cclk override field keeps the client's clock always on inside the MCCIF.

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0b00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	SATA_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	SATA_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	SATA_CCLK_OVERRIDE
17	0x0	SATA_RCLK_OVERRIDE
16	0x0	SATA_WCLK_OVERRIDE
3	DISABLE	SATA_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	SATA_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	SATA_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	SATA_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

33.9.3.26 SATA_ORDERING_RULES_0

Offset: 0x1e0 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW: Modified upstream MSIAW ordering. 0 = Tegra X1 MSIAW behavior 1 = Tegra 3 MSIAW behavior
2	0x0	UPSTREAM_RESPAW: Modified RespAW ordering. 0 = Tegra X1 RespAW behavior 1 = Tegra 3 RespAW behavior
1	0x0	UPSTREAM_RAW: Modified RAW ordering. 0 = Tegra X1 RAW behavior 1 = Tegra 3 RAW behavior

33.9.3.27 SATA_A2F_UFPCI_CFG0_0

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD

Bit	Reset	Description
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

33.9.3.28 SATA_A2F_UFPCI_CFG1_0

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

33.9.4 SATA AHCI Register Descriptions

Refer to the SATA specification for the definitions of these registers.

33.9.5 SATA and SATA0 FPCI Register Descriptions

This subsection describes the SATA and SATA0 FPCI registers. The SATA and SATA0 register sets are very similar. The register names and fields are shown as SATA<0> when both register sets are applicable.

For example, T_SATA<0>_CFG_0 defines two registers: T_SATA_CFG_0 and TSATA0_CFG_0. The register named T_SATA0_CFG_PHY_POWER is present only in the SATA0 register set.

See Chapter 2: Address Map in this TRM for the mapping of both register sets.

33.9.5.1 PCI Configuration Registers

Refer to the SATA specification for the definitions of these registers.

T_SATA<0>_CFG_0

PCI Vendor and Device ID Register

Offset: 0x00 | Read/Write: R | Reset: 0x0F70XXXX

Bit	R/W	Reset	Description
31:16	R	0F70h	T_SATA<0>_CFG_0_DEVICE_ID_UNIT: The DEVICE_ID bits identify the particular device. This identifier is allocated by the vendor. DEVICE_ID_FUNC bits contain the function number from the Configuration Address bits 10-8. DEVICE_ID_UNIT bits contain the Unit number within the chip. The Unit and Function are defined by parameters per block. D84h: DEVICE_ID_UNIT_MCP89 D84h: DEVICE_ID_UNIT_MCP89_PATA_DT D85h: DEVICE_ID_UNIT_MCP89_PATA_NB D88h: DEVICE_ID_UNIT_MCP89_AHCI_DT D89h: DEVICE_ID_UNIT_MCP89_AHCI_NB 580h: DEVICE_ID_UNIT_MCP89_AHCI_LINUX D8Ch: DEVICE_ID_UNIT_MCP89_RAID_DT D8Dh: DEVICE_ID_UNIT_MCP89_RAID_NB F70h: DEVICE_ID_UNIT_SATA (default)
15:0	R	10DEh	T_SATA<0>_CFG_0_VENDOR_ID: The VENDOR_ID bits identify the manufacturer of the device. Valid vendor identifiers are allocated by the PCI SIG to ensure uniqueness. VENDOR_ID is the joint SGS/NVIDIA PCI vendor ID. 10DEh: VENDOR_ID_NVIDIA

T_SATA<0>_CFG_1

PCI Device Control Register

The Device Control Command register provides coarse control over a device's ability to generate and respond to PCI cycles.

Offset: 0x04 | Read/Write: R/W | Reset: 0xXXXX0XXX

Bit	R/W	Reset	Description
31	R	0h	T_SATA<0>_CFG_1_DETECTED_PERR: The DETECTED_PERR bit indicates that the device has detected a parity error, even if parity error handing is disabled. (Bit 6 - T_SATA<0>_FPCI_PERR_DISABLED) 0h: DETECTED_PERR_NOT_ACTIVE
30	RW1C	0	T_SATA<0>_CFG_1_SIGNALED_SERR: The SIGNALED_SERR bit indicates that the device has asserted SERR#. NOTE: this field is reset by Cold Reset 0h: SIGNALED_SERR_NOT_ACTIVE (default) 1h: SIGNALED_SERR_ACTIVE 1h: SIGNALED_SERR_CLEAR
29	RW1C	0	T_SATA<0>_CFG_1_RECEIVED_MASTER: The RECEIVED_MASTER bit indicates that a master device's transaction (except for Special Cycle) was terminated with a master-abort. This means that no device on the PCI bus responded to the address of the mastered transaction. All master devices must implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. NOTE: this field is reset by Cold Reset 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_CLEAR
28	RW1C	0	T_SATA<0>_CFG_1_RECEIVED_TARGET: The RECEIVED_TARGET bit indicates that a master device's transaction was terminated with a target-abort. All master devices must implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. NOTE: this field is reset by Cold Reset 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_CLEAR
27	R	0h	T_SATA<0>_CFG_1_SIGNALED_TARGET: The SIGNALED_TARGET bit indicates that the device has terminated a transaction with target-abort. Devices that will never signal target-abort do not need to implement this bit. When this bit is set, an interrupt is signaled in the PBUS_INTR_0 register. 0h: SIGNALED_TARGET_NO_ABORT
26:25	R	0h	T_SATA<0>_CFG_1_DEVSEL_TIMING: The DEVSEL_TIMING bits contain the timing of DEVSEL#. There are three allowable timings for assertion of DEVSEL#. These are encoded as 00b for fast, 01b for medium, and 10b for slow (11b is reserved). These bits are read only and must indicate the slowest time that a device asserts DEVSEL# for any bus command except Configuration Read and Configuration Write. Positive decode devices are required to respond with fast DEVSEL# (0-cycle). Only the subtractive function will respond with medium DEVSEL# to accept the cycle for the subtractive bus. 0h: DEVSEL_TIMING_FAST 1h: DEVSEL_TIMING_MEDIUM 2h: DEVSEL_TIMING_SLOW
24	R	0h	T_SATA<0>_CFG_1_MASTER_DATA_PERR: 0h: MASTER_DATA_PERR_NOT_ACTIVE
23	R	1h	T_SATA<0>_CFG_1_FAST_BACK2BACK: The FAST_BACK2BACK bit indicates that the device is capable of handling back-to-back transfers when the transactions are not to the same agent. This bit can be set to 1 if the device can accept these transactions, and must be set to 0 otherwise. 1h: FAST_BACK2BACK_CAPABLE 0h: FAST_BACK2BACK_INCAPABLE
22	R	0	Reserved
21	R	1h	T_SATA<0>_CFG_1_66MHZ: The 66MHZ bit indicates that the device is capable of 66 MHz PCI Bus operation. This value is initialized by a strapping bit. 1h: 66MHZ_CAPABLE 0h: 66MHZ_INCAPABLE
20	R	1h	T_SATA<0>_CFG_1_CAPLIST: The CAPLIST bit indicates that the device configuration space includes a capabilities list starting at the offset indicated by T_SATA<0>_CFG_13. 1h: CAPLIST_PRESENT 0h: CAPLIST_NOT_PRESENT

Bit	R/W	Reset	Description
19	R	None	T_SATA<0>_CFG_1_INTR_STATUS: The INTR_STATUS bit is read-only and reflects the state of the interrupt in the device/function. Only when the INTR_DISABLE bit in the command register is a 0 and this INTR_STATUS bit is a 1, will the device's/function's INTx# signal be asserted. Setting the INTR_DISABLE bit to 1 has no effect on the state of this bit. 0h: INTR_STATUS_0 1h: INTR_STATUS_1
18:11	R	0	Reserved
10	R/W	0	T_SATA<0>_CFG_1_INTR_DISABLE: The INTR_DISABLE bit indicates that it could disable the device/function from asserting INTx#. A value of 0 enables the assertion of INTx#, and a value of 1 disables the assertion of INTx# signal. The Device Status register is used to record status information for PCI bus related events. 0h: INTR_DISABLE_ON (default) 1h: INTR_DISABLE_OFF
9	R	0h	T_SATA<0>_CFG_1_BACK2BACK: 0h: BACK2BACK_DISABLED 1h: BACK2BACK_ENABLED
8	R/W	0	T_SATA<0>_CFG_1_SERR: 0h: SERR_DISABLED (default) 1h: SERR_ENABLED
7	R	0h	T_SATA<0>_CFG_1_STEP: 0h: STEP_DISABLED 1h: STEP_ENABLED
6	R	0h	T_SATA<0>_CFG_1_PERR: 0h: PERR_DISABLED 1h: PERR_ENABLED
5	R	0h	T_SATA<0>_CFG_1_PALETTE_SNOOP: The PALETTE_SNOOP bit indicates that VGA compatible devices should snoop their palette registers. When this bit is set, special palette snooping behavior is enabled (i.e., device must not respond). When the bit is reset, the device should treat palette accesses like all other accesses. VGA compatible devices should implement this bit. PALETTE_SNOOP is writable. 0h: PALETTE_SNOOP_DISABLED 1h: PALETTE_SNOOP_ENABLED
4	R	0h	T_SATA<0>_CFG_1_WRITE_AND_INVAL: The WRITE_AND_INVAL bit indicates that the device can use the Memory Write and Invalidate command when the transfer is aligned and 16 bytes and the contents of the Cache Line Size Register is 4 DWORDS. When this bit is 1, masters may generate the command. When it is 0, Memory Write must be used instead. State after RST# is 0. This bit must be implemented by master devices that can generate the Memory Write and Invalidate command. 0h: WRITE_AND_INVAL_DISABLED 1h: WRITE_AND_INVAL_ENABLED
3	R	0h	T_SATA<0>_CFG_1_SPECIAL_CYCLE: 0h: SPECIAL_CYCLE_DISABLED 1h: SPECIAL_CYCLE_ENABLED
2	R/W	0	T_SATA<0>_CFG_1_BUS_MASTER: The BUS_MASTER bit indicates that the device can act as a master on the PCI bus. A value of 0 disables the device from generating PCI accesses. A value of 1 allows the device to behave as a bus master. BUS_MASTER is writable. 0h: BUS_MASTER_DISABLED (default) 1h: BUS_MASTER_ENABLED
1	R/W	0	T_SATA<0>_CFG_1_MEMORY_SPACE: The MEMORY_SPACE bit indicates that the device will respond to memory space accesses. A value of 0 disables the device response. A value of 1 allows the device to respond to Memory space accesses. MEMORY_SPACE is writable. 0h: MEMORY_SPACE_DISABLED (default) 1h: MEMORY_SPACE_ENABLED
0	R/W	0	T_SATA<0>_CFG_1_IO_SPACE: The IO_SPACE bit indicates that the device will respond to I/O space accesses. A value of 0 disables the device response. A value of 1 allows the device to respond to I/O space accesses. IO_SPACE is writable. 0h: IO_SPACE_DISABLED (default) 1h: IO_SPACE_ENABLED

T_SATA<0>_CFG_2

PCI Revision ID and Class Code Register

This register provides a way to configure the SATA channel to operate in either compatibility or native PCI mode.

Offset: 0x08 | Read/Write: R | Reset: 0xXXXXxxA1

Bit	R/W	Reset	Description
31:16	R	101h	T_SATA<0>_CFG_2_CLASS_CODE: The CLASS_CODE bits identify the generic function of the device and (in some cases) a specific register-level programming interface. The register is broken into three byte-size fields. The upper byte (at offset 0BH) is a base class code which broadly classifies the type of function the device performs. The middle-byte (at offset 0AH)is a sub-class code which identifies more specifically the function of the device. The lower byte (at offset 09H) identifies a specific register-level programming interface (if any) so that device independent software can interact with the device. 101h: CLASS_CODE_0101
15	R	1h	T_SATA<0>_CFG_2_SATA_BUS_MASTER: This bit defines the Bus mastering capability of SATA controller. SATA implements a back door register which can be used to change the class code. This register is T_SATA<0>_CFG_38_BKDOOR_CLASS_CODE. This provides BIOS a way to change the class code. 1h: SATA_BUS_MASTER_YES
14:12	R	0	Reserved
11	R	None (SATA) 0h (SATA0)	T_SATA<0>_CFG_2_SEC_PROG_IND: Secondary Programmable Indicator. Set to indicate it is programmable and can operate in both compatibility or native PCI mode. 0h: SEC_PROG_IND_NO (SATA0 only)
10	R	1 (SATA0)	T_SATA0_CFG_2_SEC_OP_MODE: Secondary Operating Mode 0 = Compatibility Mode (Use Legacy Addresses) 1 = Native Mode (Use Addresses Define in PCI I/O BARs) 1h: SEC_OP_MODE_INIT (default) (SATA0 only) 0h: SEC_OP_MODE_COMP 1h: SEC_OP_MODE_NTV
9	R	None (SATA) 0h (SATA0)	T_SATA<0>_CFG_2_PRI_PROG_IND: Primary Programmable Indicator. Set to indicate it is programmable and can operate in both compatibility or native PCI mode. 0h: PRI_PROG_IND_NO
8	R	1 (SATA0)	T_SATA0_CFG_2_PRI_OP_MODE: Primary Operating Mode 0 = Compatibility Mode (Use Legacy Addresses) 1 = Native Mode (Use Addresses Define in PCI I/O BARs) 1h: PRI_OP_MODE_INIT (default): (SATA0 only) 0h: PRI_OP_MODE_COMP 1h: PRI_OP_MODE_NTV
7:0	R	A1h	T_SATA<0>_CFG_2_REVISION_ID: The REVISION_ID bits specify a device specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. This field should be viewed as a vendor defined extension to the DEVICE_ID. A1h: REVISION_ID_VAL (default) A1h: REVISION_ID_A1 A2h: REVISION_ID_A2

T_SATA<0>_CFG_3

PCI Configuration Register

Offset: 0x0c | Read/Write: R/W | Reset: 0x00XXXXXX

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23	R	0h	T_SATA<0>_CFG_3_HEADER_TYPE_FUNC: 0h: HEADER_TYPE_FUNC_SINGLE 1h: HEADER_TYPE_FUNC_MULTI

Bit	R/W	Reset	Description
22:16	R	0h	T_SATA<0>_CFG_3_HEADER_TYPE_DEVICE: The HEADER_TYPE bits identify the layout of the bytes 10h: through 3FH in configuration space and also whether or not the device contains multiple functions. Bit 7 in this register is used to identify a multi-function device. If the bit is 0, then the device is single function. If the bit is 1, then the device has multiple functions. Bits 6 through 0 specify the layout of bytes 10h: through 3Fh. The LATENCY_TIMER and HEADER_TYPE are defined by parameters per block. 0h: HEADER_TYPE_DEVICE_NON_BRIDGE 1h: HEADER_TYPE_DEVICE_P2P_BRIDGE
15:11	R	0h	T_SATA<0>_CFG_3_LATENCY_TIMER: The LATENCY_TIMER bits contain, in units of PCI bus clocks, the value of the Latency Timer for this PCI bus master. This register must be implemented as writable by any master that can burst more than two data phases. This register may be implemented as read-only for devices that burst two or fewer data phases, but the hardwired value must be limited to 16 or less. A typical implementation would be to build the five high-order bits (leaving the bottom three as read-only), resulting in a timer granularity of eight clocks. At reset, the register should be set to 0 (if programmable). LATENCY_TIMER bits are writable. 0h: LATENCY_TIMER_0_CLOCKS 1h: LATENCY_TIMER_8_CLOCKS 1Eh: LATENCY_TIMER_240_CLOCKS 1Fh: LATENCY_TIMER_248_CLOCKS
10:8	R	0	Reserved
7:0	R	0h	T_SATA<0>_CFG_3_CACHE_LINE_SIZE: 0h: CACHE_LINE_SIZE_0 20h: CACHE_LINE_SIZE_32 40h: CACHE_LINE_SIZE_64

T_SATA<0>_CFG_4

PCI Configuration Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:3	R/W	0	T_SATA<0>_CFG_4_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_PRI_OP_MODE is cleared, T_SATA_CFG_4 and T_SATA_CFG_5 will always read 32'h00000000. And when T_SATA<0>_CFG_2_SEC_OP_MODE is cleared, T_SATA<0>_CFG_6 and T_SATA_CFG_7 will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_PCA
2:1	R	0	Reserved
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_4_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_5

PCI Configuration Register

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:2	R/W	0	T_SATA<0>_CFG_5_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_PRI_OP_MODE is cleared, T_SATA_CFG_4 and T_SATA_CFG_5 will always read 32'h00000000. And when T_SATA<0>_CFG_2_SEC_OP_MODE is cleared, T_SATA<0>_CFG_6 and T_SATA<0>_CFG_7 will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_PSA
1	R	0	Reserved
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_5_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_6

PCI Configuration Register

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:3	R/W	0	T_SATA<0>_CFG_6_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_PRI_OP_MODE is cleared, T_SATA_CFG_4 and T_SATA_CFG_5 will always read 32'h00000000. And when T_SATA<0>_CFG_2_SEC_OP_MODE is cleared, T_SATA<0>_CFG_6 and T_SATA_CFG_7 will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_SCA
2:1	R	0	Reserved
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_6_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_7

PCI Configuration Register

Offset: 0x1c | Read/Write: R/W | Reset: 0x0000000X (SATA) / 0x00000001 (SATA0)

Bit	R/W	Reset	Description
31:2	R/W	0	T_SATA<0>_CFG_7_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. When T_SATA<0>_CFG_2_SEC_OP_MODE is cleared, this register will always read 32'h00000000. 0h: BASE_ADDRESS_00 (default) 0h: BASE_ADDRESS_NTV_SSA
1	R	0	Reserved

Bit	R/W	Reset	Description
0	R	n/a (SATA) 1 (SATA0)	T_SATA<0>_CFG_7_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (SATA0 only) 1h: SPACE_TYPE_IO (default) (SATA0 only)

T_SATA<0>_CFG_8

PCI Configuration Register

T_SATA<0>_CFG_8 is a 16-byte I/O BAR for two things: - The Bus Master control registers for both channels.

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000000X

Bit	R/W	Reset	Description
31:4	R/W	0	T_SATA<0>_CFG_8_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device requires by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all address bits which are not required, effectively specifying the address space required. 0h: BASE_ADDRESS_BMA (default)
3:1	R	0	Reserved
0	R	1h	T_SATA<0>_CFG_8_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 1h: SPACE_TYPE_IO 0h: SPACE_TYPE_MEMORY

T_SATA<0>_CFG_9

PCI Memory BAR for AHCI Register

T_SATA<0>_CFG_9 is memory-BAR register for AHCI registers. Size of this memory space is 4KB+256Bytes. First 32 Bytes are used for Host control registers. Next 128 bytes are reserved. Next 96 bytes for vendor specific registers. Each port has 128-byte register space.

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000000X

Bit	R/W	Reset	Description
31:13	R/W	0	T_SATA<0>_CFG_9_BASE_ADDRESS: 0h: BASE_ADDRESS_INIT (default)
12:1	R	0	Reserved
0	R	0h	T_SATA<0>_CFG_9_SPACE_TYPE: 1h: SPACE_TYPE_IO 0h: SPACE_TYPE_MEMORY

T_SATA<0>_CFG_10

PCI Configuration Register

These are unused BAR locations.

Offset: 0x28 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R	0h	T_SATA<0>_CFG_10_RESERVED: 0h: RESERVED_0

T_SATA<0>_CFG_11

PCI Subsystem Vendor ID and Subsystem ID Register

Offset: 0x2c | Read/Write: R | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R	0	T_SATA<0>_CFG_11_SUBSYSTEM_ID: Subsystem ID is read-only. The system BIOS can set this value by writing to Configuration [42h]. 0h: SUBSYSTEM_ID_NONE (default)
15:0	R	0	T_SATA<0>_CFG_11_SUBSYSTEM_VENDOR_ID: Subsystem Vendor ID is read-only. The system BIOS can set this value by writing to Configuration [40h]. 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

T_SATA<0>_CFG_12

Expansion ROM Base Address Configuration Register

The Expansion ROM Base Address configuration register should not be used for any PCI integrated blocks.

Offset: 0x30 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R	0h	T_SATA<0>_CFG_12_RESERVED: 0h: RESERVED_0

T_SATA<0>_CFG_13

PCI Capability Pointer Register

Offset: 0x34 | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:0	R	44h	T_SATA<0>_CFG_13_CAP_PTR: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. This always points to 0x44 where at least the PCI-PM registers are expected to reside. 44h: CAP_PTR_PCIPM

T_SATA<0>_CFG_14

PCI Configuration Register

Offset: 0x38 | Read/Write: R | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R	0h	T_SATA<0>_CFG_14_RESERVED: The RESERVED bits are reserved for future use. 0h: RESERVED_0

T_SATA<0>_CFG_15

PCI Configuration Register

Offset: 0x3c | Read/Write: R/W | Reset: 0xXXXXXX00 (SATA) / 0xXXXX0100 (SATA0)

Bit	R/W	Reset	Description
31:24	R	1h	T_SATA<0>_CFG_15_MAX_LAT: The MAX_LAT bits contain the maximum time the device requires to gain access to the PCI bus. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MAX_LAT is nonzero. The INTR_PIN, MIN_GNT, and MAX_LAT are configurable per block. 0h: MAX_LAT_NO_REQUIREMENTS 1h: MAX_LAT_250NS

Bit	R/W	Reset	Description
23:16	R	3h	T_SATA<0>_CFG_15_MIN_GNT: The MIN_GNT bits contain the length of the burst period a device needs assuming a clock rate of 33 MHz. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MIN_GNT is nonzero. 0h: MIN_GNT_NO_REQUIREMENTS 3h: MIN_GNT_750NS
15:8	R	1 (SATA0)	T_SATA0_CFG_15_INTR_PIN: The INTR_PIN bits contain the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that don't use an interrupt pin must put a 0 in this register. 0h: INTR_PIN_NONE (SATA0 only) 1h: INTR_PIN_INTA (default) (SATA0 only) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	R/W	0	T_SATA<0>_CFG_15_INTR_LINE: The INTR_LINE bits contain the interrupt routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this register as it initializes and configures the system. The value in this register tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is initialized to 0xff (no connection) at reset. Some PCI BIOS' can't handle aliased INTR_LINEs. If T_SATA<0>_CFG_2_PRI_OP_MODE or T_SATA<0>_CFG_2_SEC_OP_MODE are set, this register will read whatever value written by software. If both the bits are cleared the register will always return 00h: 0h: INTR_LINE_IRQ0 (default) 1h: INTR_LINE_IRQ1 Fh: INTR_LINE_IRQ15 FFh: INTR_LINE_UNKNOWN

33.9.5.2 Device Specific Configuration Registers

The CFG_16 through CFG_63 registers are Device Specific PCI configuration registers.

T_SATA<0>_CFG_16

Write Subsystem Vendor ID and Subsystem ID Register

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA<0>_CFG_16_SUBSYSTEM_ID: Subsystem ID write register. The system BIOS can set the Subsystem ID value in [2Eh] by writing to this register. 0h: SUBSYSTEM_ID_NONE (default)
15:0	R/W	0	T_SATA<0>_CFG_16_SUBSYSTEM_VENDOR_ID: Subsystem Vendor ID write register. The system BIOS can set the Subsystem Vendor ID value in [2Ch] by writing to this register. 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

T_SATA<0>_FPCI_SW

Offset: 0x54 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:9	R	0h	T_SATA<0>_FPCI_SW_RSVD_9_31: 0h: RSVD_9_31_VAL
8	R/W	0	T_SATA<0>_FPCI_SW_WAKEUP_PLL: When the dev_clk PLL is shutdown, this field is written to one by SW to wakeup the PLL before issuing any other write 0h: WAKEUP_PLL_INIT (default)
7:1	R	0h	T_SATA<0>_FPCI_SW_RSVD_1_7: 0h: RSVD_1_7_VAL

Bit	R/W	Reset	Description
0	R/W	0	T_SATA<0>_FPCI_SW_IDDQ_PG: SW writes this bit to one/ zero when Power Gated which drives the IDDQ to the pads Single bit is used to drive all port's IDDQ 0h: IDDQ_PG_INIT (default)

T_SATA<0>_MSI_QUEUE

MSI Message Queue Configuration Register

The MSI_QUEUE register is a private register. It specifies to which virtual channel queue (ISO or NON-ISO) that the MSI message will be sent to before being sent out on FPCI bus.

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:4	R	0	Reserved
3:0	R/W	0	T_SATA<0>_MSI_QUEUE_Bit: This field specifies to which VC queue a particular MSI message will be sent. bit 0 corresponds to MSI vector 0 bit 1 corresponds to MSI vector 1 bit 2 corresponds to MSI vector 2 bit 3 corresponds to MSI vector 3 0h: Bit_DEFAULT (default)

T_SATA<0>_INDIRECT_IDP0

IDP pair to access 257-4K address space - Address register

Offset: 0xf0 | Read/Write: R/W | Reset: 0xXXXXX00X

Bit	R/W	Reset	Description
31:12	R	0h	T_SATA<0>_INDIRECT_IDP0_RESERVED: 0h: RESERVED_0
11:2	R/W	0	T_SATA<0>_INDIRECT_IDP0_IDP_INDEX: 0h: IDP_INDEX_0 (default)
1:0	R	0h	T_SATA<0>_INDIRECT_IDP0_RESERVED1: 0h: RESERVED1_0

T_SATA<0>_INDIRECT_IDP1

IDP pair to access 257-4K address space - DATA register

Offset: 0xf4 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:0	R/W	None	T_SATA<0>_INDIRECT_IDP1_DATA:

T_SATA<0>_FPCICFG

FPCI Debug register

The FPCICFG register is for the configuration bits that the FPCI wrapper needs.

Offset: 0xf8 | Read/Write: R/W | Reset: 0x008CC000

Bit	R/W	Reset	Description
31:28	R/W	0	T_SATA<0>_FPCICFG_DEVID_OVERRIDE_ID: Use this 4 bit register to program a desired value between 0x580-0x58F on a Linux system and between 0x550-0x55B on a no Linux system. The default values for NBP and non-NBP systems are as listed below. Non-NBP systems cannot program values of 0x2, 0x4 and 0xA. If one tried to program a value outside the limits listed here, then the last value of this field is retained. 0h: DEVID_OVERRIDE_ID_DEFAULT (default) 0h: DEVID_OVERRIDE_ID_DEFAULT_NBP

Bit	R/W	Reset	Description
27	R/W	0	T_SATA<0>_FPCICFG_DEVID_OVERRIDE_ENABLE: This is bit when set indicates a Linux system. 0h: DEVID_OVERRIDE_ENABLE_OFF (default) 0h: DEVID_OVERRIDE_ENABLE_ON
26	R/W	0	T_SATA<0>_FPCICFG_DROP_ON_TA_ERR_ENABLE: 0h: DROP_ON_TA_ERR_ENABLE_DEFAULT (default) 0h: DROP_ON_TA_ERR_ENABLE_OFF 1h: DROP_ON_TA_ERR_ENABLE_ON
25	R/W	0	T_SATA<0>_FPCICFG_DROP_ON_MA_ERR_ENABLE: 0h: DROP_ON_MA_ERR_ENABLE_DEFAULT (default) 0h: DROP_ON_MA_ERR_ENABLE_OFF 1h: DROP_ON_MA_ERR_ENABLE_ON
24	R/W	0	T_SATA<0>_FPCICFG_DROP_ON_ERR_ENABLE: 0h: DROP_ON_ERR_ENABLE_DEFAULT (default) 0h: DROP_ON_ERR_ENABLE_OFF 1h: DROP_ON_ERR_ENABLE_ON
23	R/W	1	T_SATA<0>_FPCICFG_FIX_DEADLOCK_ENABLE: 1h: FIX_DEADLOCK_ENABLE_DEFAULT (default) 0h: FIX_DEADLOCK_ENABLE_OFF 1h: FIX_DEADLOCK_ENABLE_ON
22	R/W	0	T_SATA<0>_FPCICFG_PASSIVE_UID_CLMP_ENABLE: 1h: PASSIVE_UID_CLMP_ENABLE_SET 0h: PASSIVE_UID_CLMP_ENABLE_CLR (default)
21	R	0	T_SATA<0>_FPCICFG_PASSIVE_UID_CLMP: 1h: PASSIVE_UID_CLMP_SUPP 0h: PASSIVE_UID_CLMP_NOT_SUPP 0h: PASSIVE_UID_CLMP_DEFAULT (default)
20:16	R/W	0Ch	T_SATA<0>_FPCICFG_NONISO_READ_CREDITS: Ch NONISO_READ_CREDITS_DEFAULT (default) 0h: NONISO_READ_CREDITS_MIN 1Fh NONISO_READ_CREDITS_MAX
15	R/W	1	T_SATA<0>_FPCICFG_ERR_SEVERITY: 1h: ERR_SEVERITY_DEFAULT (default) 0h: ERR_SEVERITY_NONFATAL 1h: ERR_SEVERITY_FATAL
14	R/W	1	T_SATA<0>_FPCICFG_TGTDONE_PASSPW: 1h: TGTDONE_PASSPW_DEFAULT (default) 1h: TGTDONE_PASSPW_SET 0h: TGTDONE_PASSPW_CLR
13:9	R	0	Reserved
8	R/W	0	T_SATA<0>_FPCICFG_CREDIT_SYS_ENABLE: FPCICFG_CREDIT_SYS_ENABLE is used to enable the read credit system. This is a performance feature. When this bit is set to OFF, the wrapper will assert fpci2dev_busy when the toy has reached its maximum number of outstanding NONISO or ISO read requests (determined by wrapper parameterization). When this bit is set to ON, the wrapper will only throttle toy mastered requests when there is contention for wrapper resources. This allows the toy bus master to issue writes even when the read credits have been exhausted. The T_SATA<0>_FPCICFG_ISO_READ_CREDITS field is used when the read credit system is enabled. This field sets the maximum number of outstanding ISO read requests allowed by the toy. The value in this field must NOT exceed ISO_READ_CREDITS_MAX. The T_SATA<0>_FPCICFG_NONISO_READ_CREDITS field is used when the read credit system is enabled. This field sets the maximum number of outstanding NONISO read requests allowed by the toy. The value in this field must NOT exceed NONISO_READ_CREDITS_MAX. 0h: CREDIT_SYS_ENABLE_DEFAULT (default) 0h: CREDIT_SYS_ENABLE_OFF 1h: CREDIT_SYS_ENABLE_ON
7:6	R/W	0	T_SATA<0>_FPCICFG_COHCMD: 0h: COHCMD_DEFAULT (default) 0h: COHCMD_TOY 1h: COHCMD_COH 2h: COHCMD_NONCOH

Bit	R/W	Reset	Description
5:4	R/W	0	T_SATA<0>_FPCICFG_RSPPASSPW: FPCICFG_RSPPASSPW indicates that for a master command issued from the wrapper to the UFA, it's issued with RSPPASSPW set, RSPPASSPW not set, or whatever it is sent with from the toy. These bits only influence non-posted commands. 0h: RSPPASSPW_DEFAULT (default) 0h: RSPPASSPW_TOY 1h: RSPPASSPW_PASS 2h: RSPPASSPW_NOPASS
3:2	R/W	0	T_SATA<0>_FPCICFG_PASSPW: FPCICFG_PASSPW indicates that for a master command issued from the wrapper to the UFA, it's issued with PASSPW set, PASSPW not set, or whatever it is sent with from the toy. These bits only influence non-broadcast commands. 0h: PASSPW_DEFAULT (default) 0h: PASSPW_TOY 1h: PASSPW_PASS 2h: PASSPW_NOPASS
1:0	R/W	0	T_SATA<0>_FPCICFG_ISOCMD: T_IDE_FPCICFG_ISOCMD indicates that for a master command issued from the wrapper to the UFA, it's issued as ISO, NONISO, or whatever it is sent as from the toy. These bits only influence commands that have ISO/NONISO counterparts. 0h: ISOCMD_DEFAULT (default) 0h: ISOCMD_TOY 1h: ISOCMD_ISO 2h: ISOCMD_NONISO

T_SATA<0>_SCRATCH_1

PCI Configuration Register

This general purpose scratch register is used for communication between the storage SW and the SBIOS.

This field is designed to be used to allow the SBIOS to communicate to the storage SW which ports are enabled for RAID operation.

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_SCRATCH_1_SBIOS: 0h: SBIOS_DEFAULT (default)

T_SATA<0>_NVOOB

Serial ATA internal PHY control register used in nvoob

20 + 2*ACTIVE_CNT_HIGH is the max number of 300 MHz cycles squelch can be high for OOB signaling. 20 + 2*ACTIVE_CNT_LOW is the minimum number of 300 MHz cycles squelch can be high for OOB signaling. 100 + 4*COMINIT_IDLE_CNT_HIGH is the max number of 300 MHz cycles squelch can be low for cominit signaling. 50 + 4*COMINIT_IDLE_CNT_LOW is the min number of 300 MHz cycles squelch can be low for cominit signaling. 32 + 2*COMWAKE_IDLE_CNT_HIGH is the max number of 300 MHz cycles squelch can be low for comwake signaling. 2*COMWAKE_IDLE_CNT_LOW is the min number of 300 MHz cycles squelch can be low for comwake signaling.

Offset: 0x114 | Read/Write: R/W | Reset: 0x43000000

Bit	R/W	Reset	Description
31	R/W	0	T_SATA<0>_NVOOB_RETIMED_FAREND_LOOPBACK_EN: This field when enabled puts the SATA controller in farend retimed loopback mode. 0h: RETIMED_FAREND_LOOPBACK_EN_0 (default)
30:28	R/W	4h	T_SATA<0>_NVOOB_COMMA_CNT: COMMA_CNT*16 is the number of comma characters seen for the phy_rdy to go high after going through OOB signaling. 4h: COMMA_CNT_DEFAULT (default)
27:26	R/W	3	T_SATA<0>_NVOOB_SQUELCH_FILTER_LENGTH: (SQUELCH_FILTER_LENGTH+1)*6.66ns is the amount of glitch duration that are filtered out. 3h: SQUELCH_FILTER_LENGTH_0 (default)

Bit	R/W	Reset	Description
25:24	R/W	0	T_SATA<0>_NVOOB_SQUELCH_FILTER_MODE: This field selects squelch signal filtering mode. When FILTER_MODE_LOW is selected, high-low-high glitches are filtered out. When FILTER_MODE_HIGH is selected, low-high-low glitches are filtered out. 0h: SQUELCH_FILTER_MODE_NONE (default) 1h: SQUELCH_FILTER_MODE_LOW 2h: SQUELCH_FILTER_MODE_HIGH
23:0	R	0	Reserved

T_SATA<0>_CROSS_BAR

SATA CROSS BAR

Offset: 0x118 | Read/Write: R/W | Reset: 0x00543210

Bit	R/W	Reset	Description
31:0	R/W	00543210h	T_SATA<0>_CROSS_BAR_PHY_SEL: This register contains the phy_select - indicates how a Lane (brick) is connected to the port. NOTE: this field is reset by Cold Reset 543210h: PHY_SEL_DEFAULT (default)

T_SATA<0>_PMUCTL

SATA PHY Control Register

This register contains various bits to control the PHY.

FORCE_CORE_CLAMP -- is used to enable core clock clamp. Software can only set this bit when the controller is idle. Any downstream cycle, including a read to this register, will wake up core clock automatically, and core_act_sts will change to 0x0. Software needs no future action to un-clamp the clock. But software needs to write 1 to this bit to re-clamp the clock when the controller is idle.

Note: *Wake from core clock clamp can be done by a downstream cycle which will trigger the core_wake signal to PMU; or by hotplug event which will trigger the dev_wake signal.*

DEV_STS_HOLD -- # of txclk cycles to hold the dev_act_tog and dev_act_sts. CORE_STS -- current core_act_sts. DEV_STS -- current dev_act_sts.

Offset: 0x11c | Read/Write: R/W | Reset: 0xX03F00XX

Bit	R/W	Reset	Description
31:28	R	0h	T_SATA<0>_PMUCTL_RSVD_31_28: 0h: RSVD_31_28_VAL
27:24	R	0	T_SATA<0>_PMUCTL_CORE_STS: 0h: CORE_STS_DEFAULT (default)
23:16	R/W	3Fh	T_SATA<0>_PMUCTL_DEV_STS_HOLD: NOTE: this field is reset by Cold Reset 3Fh: DEV_STS_HOLD_DEFAULT (default)
15:9	R	0	Reserved
8	R/W	0	T_SATA<0>_PMUCTL_HOLD_SEND_ALIGN_DIS: NOTE: this field is reset by Cold Reset 0h: HOLD_SEND_ALIGN_DIS_NO 1h: HOLD_SEND_ALIGN_DIS_YES 0h: HOLD_SEND_ALIGN_DIS_DEFAULT (default)
7:3	R	0h	T_SATA<0>_PMUCTL_RSVD_3_7: 0h: RSVD_3_7_VAL
2	R/W	0	T_SATA<0>_PMUCTL_FORCE_CORE_CLAMP: NOTE: this field is reset by Cold Reset 1h: FORCE_CORE_CLAMP_EN 0h: FORCE_CORE_CLAMP_DIS (default)

Bit	R/W	Reset	Description
1:0	R	0h	T_SATA<0>_PMUCTL_RSVD_0_1: 0h: RSVD_0_1_VAL

T_SATA<0>_CFG_PHY_0

Offset: 0x120 | Read/Write: R/W | Reset: 0x0000002C / 0x3228ADAC (SATA0)

Bit	R/W	Reset	Description
31:7	R	0	Reserved (SATA only)
31	R/W	0	T_SATA0_CFG_PHY_0_HOLD_RX_STAT_IDLE_IN_BIST: 0h: HOLD_RX_STAT_IDLE_IN_BIST_INIT (default)
30	R/W	0	T_SATA0_CFG_PHY_0_RESET_SREGS_EVERY_COMRESET: NOTE: this field is reset by Cold Reset 0h: RESET_SREGS_EVERY_COMRESET_INIT (default)
29	R/W	1	T_SATA0_CFG_PHY_0_ASSERT_PHYRDY_IN_NVOOB_SM: NOTE: this field is reset by Cold Reset 1h: ASSERT_PHYRDY_IN_NVOOB_SM_YES (default) 0h: ASSERT_PHYRDY_IN_NVOOB_SM_NO
28	R/W	1	T_SATA0_CFG_PHY_0_USE_STORED_COMWAKE: 1h: USE_STORED_COMWAKE_INIT (default)
27	R/W	0	T_SATA0_CFG_PHY_0_RX_STAT_IDLE: 0h: RX_STAT_IDLE_INIT (default)
26	R/W	0	T_SATA0_CFG_PHY_0_ASSERT_PHYRDY_EVEN_NOT_HRRDY: NOTE: this field is reset by Cold Reset 1h: ASSERT_PHYRDY_EVEN_NOT_HRRDY_YES 0h: ASSERT_PHYRDY_EVEN_NOT_HRRDY_NO (default)
25	R/W	1	T_SATA0_CFG_PHY_0_ASSERT_PHYRDY_FOR_BIST: NOTE: this field is reset by Cold Reset 1h: ASSERT_PHYRDY_FOR_BIST_YES (default) 0h: ASSERT_PHYRDY_FOR_BIST_NO
24	R/W	0	T_SATA0_CFG_PHY_0_MASK_SQUELCH: NOTE: this field is reset by Cold Reset 1h: MASK_SQUELCH_YES 0h: MASK_SQUELCH_NO (default)
23:22	R/W	0	T_SATA0_CFG_PHY_0_RX_SLEEP_ACTIVE: NOTE: this field is reset by Cold Reset 0h: RX_SLEEP_ACTIVE_INIT (default)
21:20	R/W	2h	T_SATA0_CFG_PHY_0_RX_SLEEP_PARTIAL: NOTE: this field is reset by Cold Reset 2h: RX_SLEEP_PARTIAL_INIT (default)
19:18	R/W	2h	T_SATA0_CFG_PHY_0_RX_SLEEP_SLUMBER: NOTE: this field is reset by Cold Reset 2h: RX_SLEEP_SLUMBER_INIT (default)
17:16	R/W	0	T_SATA0_CFG_PHY_0_TX_SLEEP_ACTIVE: NOTE: this field is reset by Cold Reset 0h: TX_SLEEP_ACTIVE_INIT (default)
15:14	R/W	2h	T_SATA0_CFG_PHY_0_TX_SLEEP_PARTIAL: NOTE: this field is reset by Cold Reset 2h: TX_SLEEP_PARTIAL_INIT (default)
13:12	R/W	2h	T_SATA0_CFG_PHY_0_TX_SLEEP_SLUMBER: NOTE: this field is reset by Cold Reset 2h: TX_SLEEP_SLUMBER_INIT (default)
11	R/W	1	T_SATA0_CFG_PHY_0_USE_7BIT_ALIGN_DET_FOR_SPD: NOTE: this field is reset by Cold Reset 1h: USE_7BIT_ALIGN_DET_FOR_SPD_YES (default) 0h: USE_7BIT_ALIGN_DET_FOR_SPD_NO
10	R/W	1	T_SATA0_CFG_PHY_0_OOB_COMINIT_UNMASK: NOTE: this field is reset by Cold Reset 1h: OOB_COMINIT_UNMASK_YES (default) 0h: OOB_COMINIT_UNMASK_NO

Bit	R/W	Reset	Description
9	R/W	0	T_SATA0_CFG_PHY_0_SEND_COMRESET_ON_WARMRESET: this bit set indicates that COMRESET will not be sent on a WARM_RESET, it will be sent only when SCTL_DET is written NOTE: this field is reset by Cold Reset 1h: SEND_COMRESET_ON_WARMRESET_YES 0h: SEND_COMRESET_ON_WARMRESET_NO (default)
8	R/W	1	T_SATA0_CFG_PHY_0_DONT_INSERT_ALIGNS_IN_BIST_L: This bit when set indicates that Aligns will not be inserted when in BIST_L mode to avoid overflow NOTE: this field is reset by Cold Reset 1h: DONT_INSERT_ALIGNS_IN_BIST_L_YES (default) 0h: DONT_INSERT_ALIGNS_IN_BIST_L_NO
7	R/W	1	T_SATA0_CFG_PHY_0_SSTS_DET_1_IN_PART_SLUMBER: This bit when set indicates that ssts_det will be made one when we go to PARTIAL/ SLUMBER, else zero NOTE: this field is reset by Cold Reset 1h: SSTS_DET_1_IN_PART_SLUMBER_YES (default) 0h: SSTS_DET_1_IN_PART_SLUMBER_NO
6	R/W	0	T_SATA<0>_CFG_PHY_0_PLL_IDDQ_OVERRIDE_VAL: NOTE: this field is reset by Cold Reset 1h: PLL_IDDQ_OVERRIDE_VAL_YES 0h: PLL_IDDQ_OVERRIDE_VAL_NO (default)
5	R/W	1	T_SATA<0>_CFG_PHY_0_PLL_IDDQ_OVERRIDE: NOTE: this field is reset by Cold Reset 1h: PLL_IDDQ_OVERRIDE_YES (default) 0h: PLL_IDDQ_OVERRIDE_NO
4:0	R/W	0Ch	T_SATA<0>_CFG_PHY_0_RBC_RESET_DELAY: RBC_RESET_DELAY is the number of tx_clk ticks (300MHz) between the deassertion of sata2phy_ch{1,2}_cdr_reset and the release of ch{1,2}_rbc_reset_. NOTE: this field is reset by Cold Reset Ch: RBC_RESET_DELAY_DEFAULT (default)

T_SATA0_CFG_PHY_POWER

This register applies to SATA0 register space only.

Offset: 0x124 | Read/Write: R/W | Reset: 0x96EA6001

Bit	R/W	Reset	Description
31:24	R/W	96h	T_SATA0_CFG_PHY_POWER_COUNT_FOR_1_US: 96h: COUNT_FOR_1_US_INIT (default)
23:10	R/W	3A98h	T_SATA0_CFG_PHY_POWER_COUNT_FOR_100_US: 3A98h: COUNT_FOR_100_US_INIT (default)
9:0	R/W	1	T_SATA0_CFG_PHY_POWER_TIME_WAIT_IN_PARTIAL: indicates the time in ms up to which the PHY SM would wait in PARTIAL before going to SLUMBER(AUTO PARTIAL TO SLUMBER feature) 1h: TIME_WAIT_IN_PARTIAL_INIT (default)

T_SATA0_CFG_PHY_POWER_1

This register applies to SATA0 register space only.

Offset: 0x128 | Read/Write: R/W | Reset: 0x0FF249F0

Bit	R/W	Reset	Description
31:20	R/W	0FFh	T_SATA0_CFG_PHY_POWER_1_COUNTER_FOR_PADS: FFh: COUNTER_FOR_PADS_INIT (default)
19:0	R/W	249F0h	T_SATA0_CFG_PHY_POWER_1_COUNT_FOR_1_MS: Indicates the number of clock cycles for one ms at 150 MHz (sata_oob_detect_clk). Used for Auto Partial to slumber feature 249F0h: COUNT_FOR_1_MS_INIT (default)

T_SATA<0>_CFG_PHY_1

Offset: 0x12c | Read/Write: R/W | Reset: 0x0732400F

Bit	R/W	Reset	Description
31:27	R	0	Reserved
26	R/W	1	T_SATA<0>_CFG_PHY_1_DONT_CHK_PHY_RESET: NOTE: this field is reset by Cold Reset 1h: DONT_CHK_PHY_RESET_INIT (default)
25	R/W	1	T_SATA<0>_CFG_PHY_1_COMWAKE_GLOBAL: NOTE: this field is reset by Cold Reset 1h: COMWAKE_GLOBAL_INIT (default)
24	R/W	1	T_SATA<0>_CFG_PHY_1_PLL_PD_NO_CMDS: 1h: PLL_PD_NO_CMDS_INIT (default)
23	R/W	0	T_SATA<0>_CFG_PHY_1_PADS_IDDQ_EN: NOTE: this field is reset by Cold Reset 0h: PADS_IDDQ_EN_INIT (default)
22	R/W	0	T_SATA<0>_CFG_PHY_1_PAD_PLL_IDDQ_EN: NOTE: this field is reset by Cold Reset 0h: PAD_PLL_IDDQ_EN_INIT (default)
21	R/W	1	T_SATA<0>_CFG_PHY_1_SEND_OOB_DATA_IN_LOW_POWER: NOTE: this field is reset by Cold Reset 1h: SEND_OOB_DATA_IN_LOW_POWER_INIT (default)
20	R/W	1	T_SATA<0>_CFG_PHY_1_NO_OVERRIDE_STAT_IDLE_PHYRDY: NOTE: this field is reset by Cold Reset 1h: NO_OVERRIDE_STAT_IDLE_PHYRDY_INIT (default)
19:16	R/W	2h	T_SATA<0>_CFG_PHY_1_NUMBER_OF_COMMA_WINDOWS:
15:4	R/W	400h	T_SATA<0>_CFG_PHY_1_COUNT_FOR_COMMA_WAIT: NOTE: this field is reset by Cold Reset 400h: COUNT_FOR_COMMA_WAIT_INIT (default)
3	R/W	1	T_SATA<0>_CFG_PHY_1_DONT_USE_COMMA_FOR_PHYRDY_LOW: NOTE: this field is reset by Cold Reset 1h: DONT_USE_COMMA_FOR_PHYRDY_LOW_INIT (default)
2	R/W	1	T_SATA<0>_CFG_PHY_1_EN_ASYNC_REC_ARC_IN_HRRDY: NOTE: this field is reset by Cold Reset 1h: EN_ASYNC_REC_ARC_IN_HRRDY_INIT (default)
1	R/W	1	T_SATA<0>_CFG_PHY_1_HOLD_RBC_RESET_IN_BIST: 1h: HOLD_RBC_RESET_IN_BIST_INIT (default)
0	R/W	1	T_SATA<0>_CFG_PHY_1_ASSERT_PHYRDY_FOR_ALL_BIST: 1h: ASSERT_PHYRDY_FOR_ALL_BIST_INIT (default)

T_SATA<0>_CFG2NVOOB_1

Offset: 0x130 | Read/Write: R/W | Reset: 0x01802220

Bit	R/W	Reset	Description
31:27	R	0	Reserved
26:18	R/W	060h	T_SATA<0>_CFG2NVOOB_1_COMINIT_IDLE_CNT_HIGH: Squelch can be high for a maximum of COMINIT_IDLE_CNT_HIGH cycles at 216MHz for COMINIT signaling. NOTE: this field is reset by Cold Reset 60h: COMINIT_IDLE_CNT_HIGH_DEFAULT (default)
17:9	R/W	011h	T_SATA<0>_CFG2NVOOB_1_ACTIVE_CNT_LOW: Squelch can be high for a minimum of ACTIVE_CNT_LOW cycles at 216MHz for OOB signaling. NOTE: this field is reset by Cold Reset 11h: ACTIVE_CNT_LOW_DEFAULT (default)
8:0	R/W	020h	T_SATA<0>_CFG2NVOOB_1_ACTIVE_CNT_HIGH: Squelch can be high for a maximum of ACTIVE_CNT_HIGH cycles at 216MHz for OOB signaling. NOTE: this field is reset by Cold Reset 20h: ACTIVE_CNT_HIGH_DEFAULT (default)

T_SATA<0>_CFG2NVOOB_2

Offset: 0x134 | Read/Write: R/W | Reset: 0x00444C38

Bit	R/W	Reset	Description
31:27	R	0	Reserved
26:18	R/W	011h	T_SATA<0>_CFG2NVOOB_2_COMWAKE_IDLE_CNT_LOW: Squelch can be high for a minimum of 2*COMWAKE_IDLE_CNT_LOW cycles at 216MHz for COMWAKE signaling. NOTE: this field is reset by Cold Reset 11h: COMWAKE_IDLE_CNT_LOW_DEFAULT (default)
17:9	R/W	026h	T_SATA<0>_CFG2NVOOB_2_COMWAKE_IDLE_CNT_HIGH: Squelch can be high for a maximum of 32 + 2*COMWAKE_IDLE_CNT_HIGH cycles at 216MHz for COMWAKE signaling. NOTE: this field is reset by Cold Reset 26h: COMWAKE_IDLE_CNT_HIGH_DEFAULT (default)
8:0	R/W	038h	T_SATA<0>_CFG2NVOOB_2_COMINIT_IDLE_CNT_LOW: Squelch can be high for a minimum of 50 + 4*COMINIT_IDLE_CNT_LOW cycles at 216MHz for COMINIT signaling. NOTE: this field is reset by Cold Reset 38h: COMINIT_IDLE_CNT_LOW_DEFAULT (default)

T_SATA<0>_CFG_PHY_ACTIVE

Offset: 0x138 | Read/Write: R/W | Reset: 0x0001705C

Bit	R/W	Reset	Description
31:10	R/W	00005Ch	T_SATA<0>_CFG_PHY_ACTIVE_FROM_SLUMBER: Decides the time to be in SLUMEBR state after we receive a COMWAKE from the device or the Link Layer lowers the slumber flag. To give time for the pads to wakeup before sending the COMWAKE. This is for txclk of 300MHZ, will be multiplied by two in the OOB state-machine for 600MHz Changing the default value of SLUMBER to 0x5C since Partial and Slumber wakeup times are the same. NOTE: this field is reset by Cold Reset 3AA8h: FROM_SLUMBER_DEFAULT_1 5Ch: FROM_SLUMBER_DEFAULT (default)
9:0	R/W	05Ch	T_SATA<0>_CFG_PHY_ACTIVE_FROM_PARTIAL: Decides the time to be in PARTIAL state after we receive a COMWAKE from the device or the Link Layer lowers the partial flag. To give time for the pads to wakeup before sending the COMWAKE. This is for txclk of 300MHZ, will be multiplied by two in the OOB state-machine for 600MHz NOTE: this field is reset by Cold Reset 5Ch: FROM_PARTIAL_DEFAULT (default)

T_SATA<0>_FIFO

Offset: 0x170 | Read/Write: R/W | Reset: 0x00086000

Bit	R/W	Reset	Description
31:20	R	0	Reserved
19:16	R/W	8h	T_SATA<0>_FIFO_P2L_FIFO_DEPTH: This controls effective depth of p2l FIFO. Smaller is better for performance but we can't do power optimization (clamping fpci_clk) with smaller depth FIFOs. NOTE: this field is reset by Cold Reset 8h: P2L_FIFO_DEPTH_DEFAULT (default)
15:12	R/W	6h	T_SATA<0>_FIFO_L2P_FIFO_DEPTH: This controls effective depth of l2p FIFO. Smaller is better for performance but we can't do power optimization (clamping fpci_clk) with smaller depth FIFOs. NOTE: this field is reset by Cold Reset 6h: L2P_FIFO_DEPTH_DEFAULT (default)
11:0	R	0	Reserved

T_SATA<0>_CFG_LINK_0

Offset: 0x174 | Read/Write: R/W | Reset: 0xXX00009A (SATA) / 0xXXDD239A (SATA0)

Bit	R/W	Reset	Description
31:25	R	0h	T_SATA<0>_CFG_LINK_0_RSVD: 0h: RSVD_VAL
24:8	0	R	Reserved (SATA only)
24	R/W	0	T_SATA0_CFG_LINK_0_USE_POSEDGE_SCTL_DET: NOTE: this field is reset by Cold Reset 0h: USE_POSEDGE_SCTL_DET_INIT (default)
23	R/W	1	T_SATA0_CFG_LINK_0_USE_DET_OR_PSM2LL_RESET: NOTE: this field is reset by Cold Reset 1h: USE_DET_OR_PSM2LL_RESET_INIT (default)
22	R/W	1	T_SATA0_CFG_LINK_0_HARDCODE_DISPARITY_ON_WAKE: NOTE: this field is reset by Cold Reset 1h: HARDCODE_DISPARITY_ON_WAKE_YES (default) 0h: HARDCODE_DISPARITY_ON_WAKE_NO
21	R/W	0	T_SATA0_CFG_LINK_0_USE_IS_PRIM_FOR_BIST: 0h: USE_IS_PRIM_FOR_BIST_INIT (default)
20	R/W	1	T_SATA0_CFG_LINK_0_WAIT_FOR_PSM_FOR_PMOFF: NOTE: this field is reset by Cold Reset 1h: WAIT_FOR_PSM_FOR_PMOFF_INIT (default)
19	R/W	1	T_SATA0_CFG_LINK_0_USE_AHCI_MODE_FOR_COMRESET: NOTE: this field is reset by Cold Reset 1h: USE_AHCI_MODE_FOR_COMRESET_YES (default)
18	R/W	1	T_SATA0_CFG_LINK_0_GOTO_WAKE_UP4: NOTE: this field is reset by Cold Reset 1h: GOTO_WAKE_UP4_YES (default) 0h: GOTO_WAKE_UP4_NO
17	R/W	0	T_SATA0_CFG_LINK_0_DONT_CHK_PHYRDY_IN_NO_COMM: NOTE: this field is reset by Cold Reset 1h: DONT_CHK_PHYRDY_IN_NO_COMM_YES 0h: DONT_CHK_PHYRDY_IN_NO_COMM_NO (default)
16	R/W	1	T_SATA0_CFG_LINK_0_SEND_NEG_RD_ALIGNS: NOTE: this field is reset by Cold Reset 1h: SEND_NEG_RD_ALIGNS_YES (default) 0h: SEND_NEG_RD_ALIGNS_NO
15	R/W	0	T_SATA0_CFG_LINK_0_OLD_BEHAVIOUR_SEND_ALIGNS: NOTE: this field is reset by Cold Reset 1h: OLD_BEHAVIOUR_SEND_ALIGNS_YES 0h: OLD_BEHAVIOUR_SEND_ALIGNS_NO (default)
14:10	R/W	08h	T_SATA0_CFG_LINK_0_PM_OFF_COUNT: Indicates the number of PMREQ send to the device on Slumber/ PARTIAL NOTE: this field is reset by Cold Reset 8h: PM_OFF_COUNT_DEFAULT (default)
9	R/W	1	T_SATA0_CFG_LINK_0_SEND_RAW_DATA_IN_BIST_L: Default to zero, when set sends raw data in BIST_L mode without encoding/ decoding NOTE: this field is reset by Cold Reset 1h: SEND_RAW_DATA_IN_BIST_L_YES (default) 0h: SEND_RAW_DATA_IN_BIST_L_NO
8	R/W	1	T_SATA0_CFG_LINK_0_BE_IN_BIST_ON_PHYRDY_LOW: The default value of this bit is one, Host should come out of BIST only on COMINIT/ COMRESET, default to that NOTE: this field is reset by Cold Reset 1h: BE_IN_BIST_ON_PHYRDY_LOW_YES (default) 0h: BE_IN_BIST_ON_PHYRDY_LOW_NO
7	R/W	1	T_SATA<0>_CFG_LINK_0_GOTO_SEND_SYNC_P: This bit set sends Link SM to L_SEND_SYNC_P in BIST NOTE: this field is reset by Cold Reset 0h: GOTO_SEND_SYNC_P_NO 1h: GOTO_SEND_SYNC_P_YES (default)

Bit	R/W	Reset	Description
6	R/W	0	T_SATA<0>_CFG_LINK_0_AUTO_REPEAT_PRIMS: NOTE: this field is reset by Cold Reset 0h: AUTO_REPEAT_PRIMS_NO 1h: AUTO_REPEAT_PRIMS_YES 0h: AUTO_REPEAT_PRIMS_INIT (default)
5	R/W	0	T_SATA<0>_CFG_LINK_0_DEBOUNCE_PHYRDY_PERIOD: When set to 1, the hysteresis period is let to LONG mode. A setting of zero is SHORT mode. NOTE: this field is reset by Cold Reset 1h: DEBOUNCE_PHYRDY_PERIOD_LONG 0h: DEBOUNCE_PHYRDY_PERIOD_SHORT 0h: DEBOUNCE_PHYRDY_PERIOD_INIT (default)
4	R/W	1	T_SATA<0>_CFG_LINK_0_DEBOUNCE_PHYRDY: This bit and next bit controls phyrdy debounce behavior. By default phyrdy is debounced. Once bit this is set, bit 5 selects the period of hysteresis. NOTE: this field is reset by Cold Reset 1h: DEBOUNCE_PHYRDY_YES 0h: DEBOUNCE_PHYRDY_NO 1h: DEBOUNCE_PHYRDY_INIT (default)
3	R/W	1	T_SATA<0>_CFG_LINK_0_DELAY_HOTPLUG_INTR: This bit is used to control generation of hotplug interrupts. If this bit is set to 1, then an interrupt is generated when we get PHYRDY. If this bit is set to 0, then an interrupt is generated as soon as cominit is received. NOTE: this field is reset by Cold Reset 1h: DELAY_HOTPLUG_INTR_YES (default) 0h: DELAY_HOTPLUG_INTR_NO
2	R/W	0	T_SATA<0>_CFG_LINK_0_SET_BSY_BIT_MTHD: This controls when 8'h80 is loaded into shadow status reg. If the bit is 0 then 8'h80 is loaded when cominit is seen otherwise its loaded when phyrdy is seen. NOTE: this field is reset by Cold Reset 1h: SET_BSY_BIT_MTHD_PHYRDY 0h: SET_BSY_BIT_MTHD_COMINIT (default)
1:0	R/W	2h	T_SATA<0>_CFG_LINK_0_LED_MIN_ON_TIME: Used to know the activity based on LED blinking NOTE: this field is reset by Cold Reset 0h: LED_MIN_ON_TIME_OFF 1h: LED_MIN_ON_TIME_20MS 2h: LED_MIN_ON_TIME_40MS (default) 3h: LED_MIN_ON_TIME_80MS

T_SATA<0>_CFG_LINK_1

Offset: 0x178 | Read/Write: R/W | Reset: 0x02000200

Bit	R/W	Reset	Description
31:16	R/W	0200h	T_SATA<0>_CFG_LINK_1_GEN3_DWRD_WAIT_CNT: This field is the number of generation3 ALIGN dwords the controller sends to drive while waiting for gen3 align from drive, before it gives up and switches to gen2 mode of operation. It should be set to a value smaller than 2*2048. It should also be divisible by 2. NOTE: this field is reset by Cold Reset 200h: GEN3_DWRD_WAIT_CNT_INIT (default)
15:0	R/W	0200h	T_SATA<0>_CFG_LINK_1_GEN2_DWRD_WAIT_CNT: This field is the number of generation2 ALIGN dwords the controller sends to drive while waiting for gen2 align from drive, before it gives up and switches to gen1 mode of operation. It should be set to a value smaller than 2*2048. It should also be divisible by 2. NOTE: this field is reset by Cold Reset 200h: GEN2_DWRD_WAIT_CNT_INIT (default)

T_SATA<0>_CFG_LINK_2

Offset: 0x17c | Read/Write: R/W | Reset: 0x00007080

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:12	R/W	7h	T_SATA<0>_CFG_LINK_2_PHYRDY_USE_ITH_BIT: NOTE: this field is reset by Cold Reset 7h: PHYRDY_USE_ITH_BIT_INIT (default)

Bit	R/W	Reset	Description
11	R/W	0	T_SATA<0>_CFG_LINK_2_USE_BIT_FOR_DEBOUNCE: NOTE: this field is reset by Cold Reset 0h: USE_BIT_FOR_DEBOUNCE_INIT (default)
10:0	R/W	080h	T_SATA<0>_CFG_LINK_2_PHYRDY_DBNC_MUX: We have a debounced version of PHYRDY which has two options 10 ms and 10 µs (for a clock of around 100MHz - devclk) NOTE: this field is reset by Cold Reset 80h: PHYRDY_DBNC_MUX_INIT (default)

T_SATA<0>_CFG_TRANS_0

Offset: 0x1d0 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:2	R	0h	T_SATA<0>_CFG_TRANS_0_RSVD: 0h: RSVD_VAL
1	R/W	1	T_SATA<0>_CFG_TRANS_0_USE_RISE_EDGE_STATUS_RESET: NOTE: this field is reset by Cold Reset 1h: USE_RISE_EDGE_STATUS_RESET_YES (default) 0h: USE_RISE_EDGE_STATUS_RESET_NO
0	R/W	0	T_SATA<0>_CFG_TRANS_0_F2I_FIFO_FLUSH_FIX: NOTE: this field is reset by Cold Reset 1h: F2I_FIFO_FLUSH_FIX_YES 0h: F2I_FIFO_FLUSH_FIX_NO (default)

T_SATA0_ALPM_CTRL

ALPM Controls

This register applies to SATA0 register space only.

Offset: 0x238 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA0_ALPM_CTRL_THRESHOLD: This field is used to indicate to HBA as to how much time it should wait before going to Partial or Slumber power down states. It is used to load the 16-bit start value of a down-counter in RTL. Default value zero indicates that if PxCI and PxSACT registers are cleared to zero (implies that if there are no commands outstanding) the HBA SM will transition to initiate power down. NOTE: this field is reset by Cold Reset 0h: THRESHOLD_DEFAULT (default)
15:0	R	0	Reserved

T_SATA0_FBS_CONFIG_0

This register implements settings for FIS based switching implementation.

This register applies to SATA0 register space only.

Offset: 0x23c | Read/Write: R/W | Reset: 0xF0F0DFF9

Bit	R/W	Reset	Description
31:16	R/W	F0F0h	T_SATA0_FBS_CONFIG_0_CTL_03_RSVD: F0F0h: CTL_03_RSVD_INIT (default)
15:14	R/W	3h	T_SATA0_FBS_CONFIG_0_PRD_PROCESS_FIX: 3h: PRD_PROCESS_FIX_INIT (default)
13	R/W	0	T_SATA0_FBS_CONFIG_0_POST_1ST_REGFIS: 0h: POST_1ST_REGFIS_INIT (default)
12	R/W	1	T_SATA0_FBS_CONFIG_0_CTL_02: 1h: CTL_02_INIT (default)
11:8	R/W	Fh	T_SATA0_FBS_CONFIG_0_BKDR_ADO: Fh: BKDR_ADO_INIT (default)

Bit	R/W	Reset	Description
7:4	R/W	Fh	T_SATA0_FBS_CONFIG_0_CONTROL_PORT: Fh: CONTROL_PORT_INIT (default)
3	R/W	1	T_SATA0_FBS_CONFIG_0_CONTROL_PORT_HAS_PRIORITY: 1h: CONTROL_PORT_HAS_PRIORITY_INIT (default)
2	R/W	0	T_SATA0_FBS_CONFIG_0_CTL_02_RSVD: 0h: CTL_02_RSVD_INIT (default)
1	R/W	0	T_SATA0_FBS_CONFIG_0_CTL_01: 0h: CTL_01_INIT (default)
0	R/W	1	T_SATA0_FBS_CONFIG_0_CTL_00: 1h: CTL_00_INIT (default)

T_SATA0_CHX_FBS_CONFIG_1

This register implements settings for FIS based switching implementation.

This register applies to SATA0 register space only.

Offset: 0x240 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:1	R	0h	T_SATA0_CHX_FBS_CONFIG_1_RSVD_1_31: 0h: RSVD_1_31_VAL
0	R/W	0	T_SATA0_CHX_FBS_CONFIG_1_PORT_BKDR_FBSCP: NOTE: this field is reset by Cold Reset 0h: PORT_BKDR_FBSCP_INIT (default)

T_SATA0_AHCI_HBA_CAP_BKDR

This register will have a backdoor field for every bit described in the AHCI 1.2 specification.

This register applies to SATA0 register space only.

Offset: 0x300 | Read/Write: R/W | Reset: 0xE620FF01

Bit	R/W	Reset	Description
31	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_S64A: 1h: S64A_TRUE (default) 0h: S64A_FALSE
30	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SNCQ: 1h: SNCQ_TRUE (default) 0h: SNCQ_FALSE
29	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SSNTF: 1h: SSNTF_TRUE (default) 0h: SSNTF_FALSE
28	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SMPS: 1h: SMPS_TRUE 0h: SMPS_FALSE (default)
27	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_STG_SPUP: Backdoor field to advertise staggered spin up. BIOS has to write this we are going to support staggered spin up. 1h: SUPP_STG_SPUP_SET 0h: SUPP_STG_SPUP_CLEAR (default)
26	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SALP: 1h: SALP_TRUE (default) 0h: SALP_FALSE
25	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SAL: 1h: SAL_TRUE (default) 0h: SAL_FALSE
24	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_CLO: 1h: SUPP_CLO_TRUE 0h: SUPP_CLO_FALSE (default)

Bit	R/W	Reset	Description
23:20	R/W	2h	T_SATA0_AHCI_HBA_CAP_BKDR_INTF_SPD_SUPP: 0h: INTF_SPD_SUPP_RSVD 1h: INTF_SPD_SUPP_GEN1 2h: INTF_SPD_SUPP_GEN1_2 (default) 3h: INTF_SPD_SUPP_GEN3
19	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_NONZERO_OFFSET: 1h: SUPP_NONZERO_OFFSET_TRUE 0h: SUPP_NONZERO_OFFSET_FALSE (default)
18	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_AHCI_ONLY: 1h: SUPP_AHCI_ONLY_TRUE 0h: SUPP_AHCI_ONLY_FALSE (default)
17	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_SUPP_PM: 1h: SUPP_PM_TRUE 0h: SUPP_PM_FALSE (default)
16	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_FIS_SWITCHING: 1h: FIS_SWITCHING_TRUE 0h: FIS_SWITCHING_FALSE (default)
15	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_PIO_MULT_DRQ_BLK: 1h: PIO_MULT_DRQ_BLK_SUPP (default) 0h: PIO_MULT_DRQ_BLK_NOT_SUPP
14	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_SLUMBER_ST_CAP: 1h: SLUMBER_ST_CAP_TRUE (default) 0h: SLUMBER_ST_CAP_FALSE
13	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_PARTIAL_ST_CAP: 1h: PARTIAL_ST_CAP_TRUE (default) 0h: PARTIAL_ST_CAP_FALSE
12:8	R/W	1Fh	T_SATA0_AHCI_HBA_CAP_BKDR_NUM_CMD_SLOTS: 0h: NUM_CMD_SLOTS_1 7h: NUM_CMD_SLOTS_8 Fh: NUM_CMD_SLOTS_16 1Fh: NUM_CMD_SLOTS_32 (default)
7	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_CMD_CMPL_COALESING: 1h: CMD_CMPL_COALESING_TRUE 0h: CMD_CMPL_COALESING_FALSE (default)
6	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_ENCL_MGMT_SUPP: 1h: ENCL_MGMT_SUPP_TRUE 0h: ENCL_MGMT_SUPP_FALSE (default)
5	R/W	0	T_SATA0_AHCI_HBA_CAP_BKDR_EXT_SATA: This is a backdoor access for enabling external SATA. 1h: EXT_SATA_SUPPORTED 0h: EXT_SATA_NOT_SUPPORTED (default)
4:0	R/W	1	T_SATA0_AHCI_HBA_CAP_BKDR_NUM_PORTS: 0h: NUM_PORTS_1 1h: NUM_PORTS_2 2h: NUM_PORTS_3 3h: NUM_PORTS_4 1h: NUM_PORTS_DEFUALT (default)

T_SATA0_AHCI_HBA_HOLD_GEN

This register applies to SATA0 register space only.

Offset: 0x304 | Read/Write: R/W | Reset: 0x19080026

Bit	R/W	Reset	Description
31:24	R/W	19h	T_SATA0_AHCI_HBA_HOLD_GEN_SHARED_DFIFO_AVAIL: This field is programmed with a variable water mark for generation of HOLD to drive depending on depth of shared DATA FIFO. This field indicates the number of lines of the FIFO and not the size in bytes/words. NOTE: this field is reset by Cold Reset 19h: SHARED_DFIFO_AVAIL_INIT (default)

Bit	R/W	Reset	Description
23:8	R/W	0800h	T_SATA0_AHCI_HBA_HOLD_GEN_PRD_SIZE: This field indicates the minimum value of PRDBC for the last PRD cached below which Port should send a HOLD to drive. NOTE: this field is reset by Cold Reset 800h: PRD_SIZE_INIT (default)
7:0	R/W	26h	T_SATA0_AHCI_HBA_HOLD_GEN_T2P_FIFO_AVAIL: This field is programmed with a variable watermark for generation of HOLD to drive depending on the depth of T2P FIFO. This field indicates the number of lines of the FIFO and not the size in bytes/ words. NOTE: this field is reset by Cold Reset 26h: T2P_FIFO_AVAIL_INIT (default) 2Eh: T2P_FIFO_AVAIL_MAX

T_SATA0_AHCI_HBA_CTL_0

This register applies to SATA0 register space only.

Offset: 0x30c | Read/Write: R/W | Reset: 0x7C070014

Bit	R/W	Reset	Description
31	R/W	0	T_SATA0_AHCI_HBA_CTL_0_USE_PXCMD_ICC_IF_LINK_IN_IDLE: 1h: USE_PXCMD_ICC_IF_LINK_IN_IDLE_YES 0h: USE_PXCMD_ICC_IF_LINK_IN_IDLE_NO (default)
30	R/W	1	T_SATA0_AHCI_HBA_CTL_0_USE_AHCI_MODE_IN_FIS_PROCESS: 1h: USE_AHCI_MODE_IN_FIS_PROCESS_YES (default) 0h: USE_AHCI_MODE_IN_FIS_PROCESS_NO
29:28	R/W	3h	T_SATA0_AHCI_HBA_CTL_0_BKDR_VS_MINOR_9_8: 3h: BKDR_VS_MINOR_9_8_DEFAULT (default)
27	R/W	1	T_SATA0_AHCI_HBA_CTL_0_TRANSPRT_IGNORE_DMAT: 1h: TRANSPRT_IGNORE_DMAT_TRUE (default) 0h: TRANSPRT_IGNORE_DMAT_FALSE
26	R/W	1	T_SATA0_AHCI_HBA_CTL_0_PSM2LL_DENY_PMREQ: 1h: PSM2LL_DENY_PMREQ_TRUE (default) 0h: PSM2LL_DENY_PMREQ_FALSE
25	R/W	0	T_SATA0_AHCI_HBA_CTL_0_SUD_OLD_LOGIC: NOTE: this field is reset by Cold Reset 1h: SUD_OLD_LOGIC_TRUE 0h: SUD_OLD_LOGIC_FALSE (default)
24	R/W	0	T_SATA0_AHCI_HBA_CTL_0_RSVD_24: NOTE: this field is reset by Cold Reset 0h: RSVD_24_ZERO (default)
23:20	R/W	0	T_SATA0_AHCI_HBA_CTL_0_ACCEL_SYNC_ESC_TIMEOUT: 0h: ACCEL_SYNC_ESC_TIMEOUT_DEFAULT (default)
19	R/W	0	T_SATA0_AHCI_HBA_CTL_0_RST_AE_WITHOUT_COMRESET: NOTE: this field is reset by Cold Reset 0h: RST_AE_WITHOUT_COMRESET_DEFAULT (default)
18	R/W	1	T_SATA0_AHCI_HBA_CTL_0_RST_AE_ON_HR: NOTE: this field is reset by Cold Reset 1h: RST_AE_ON_HR_DEFAULT (default)
17	R/W	1	T_SATA0_AHCI_HBA_CTL_0_HBFS_ON_PRD_ADR_EXCEED_40B: 1h: HBFS_ON_PRD_ADR_EXCEED_40B_INIT (default)
16	R/W	1	T_SATA0_AHCI_HBA_CTL_0_HBFS_ON_BME_RESET_WHILE_ACTIVE: 1h: HBFS_ON_BME_RESET_WHILE_ACTIVE_DEFAULT (default)
15:8	R	0	Reserved
7	R/W	0	T_SATA0_AHCI_HBA_CTL_0_ACCEL_WAIT_FOR_BSY0: 1h: ACCEL_WAIT_FOR_BSY0_ENABLE 0h: ACCEL_WAIT_FOR_BSY0_DISABLE (default)
6	R/W	0	T_SATA0_AHCI_HBA_CTL_0_NO_DHRS_AT_UNLOAD: 1h: NO_DHRS_AT_UNLOAD_ENABLE 0h: NO_DHRS_AT_UNLOAD_DISABLE (default)
5	R/W	0	T_SATA0_AHCI_HBA_CTL_0_MPIS_SET_AT_ACCEL: 1h: MPIS_SET_AT_ACCEL_ENABLE 0h: MPIS_SET_AT_ACCEL_DISABLE (default)

Bit	R/W	Reset	Description
4	R/W	1	T_SATA0_AHCI_HBA_CTL_0_DIS_CCC_TIMEOUT_INT_AT_ZERO_OUTSTD_CMD: 1h: DIS_CCC_TIMEOUT_INT_AT_ZERO_OUTSTD_CMD_SET (default) 0h: DIS_CCC_TIMEOUT_INT_AT_ZERO_OUTSTD_CMD_CLEAR
3	R/W	0	T_SATA0_AHCI_HBA_CTL_0_BKDR_PRDBC_UPDATE: AHCI spec says that we need to update PRDBC before clearing PXCI for a non-native command. And optionally may update after each data FIS completion. 1h: BKDR_PRDBC_UPDATE_AT_EACH_DATA_FIS 0h: BKDR_PRDBC_UPDATE_AT_REG_FIS (default)
2	R/W	1	T_SATA0_AHCI_HBA_CTL_0_PORTX_NONNCQ_TX_RX_UNDERFLOW: 0h: PORTX_NONNCQ_TX_RX_UNDERFLOW_DISABLE 1h: PORTX_NONNCQ_TX_RX_UNDERFLOW_ENABLE (default)
1	R/W	0	T_SATA0_AHCI_HBA_CTL_0_PORTX_PRD_UNDERFLOW_IFS_ATAPI: 0h: PORTX_PRD_UNDERFLOW_IFS_ATAPI_DISABLE (default) 1h: PORTX_PRD_UNDERFLOW_IFS_ATAPI_ENABLE
0	R/W	0	T_SATA0_AHCI_HBA_CTL_0_PORTX_OFS_IN_ATAPI: 0h: PORTX_OFS_IN_ATAPI_DISABLE (default) 1h: PORTX_OFS_IN_ATAPI_ENABLE

T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL

This register has overrides for various BIST modes available according to the specification

This register applies to SATA0 register space only.

Offset: 0x318 | Read/Write: R/W | Reset: 0x00000600T_SATA0_AHCI_HBA_BIST_DWORD

Bit	R/W	Reset	Description
31:24	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_PORTS: 0h: PORTS_DEFAULT (default)
23:11	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_RSVD: 0h: RSVD_DEFAULT (default)
10	R/W	1	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_DISABLE_ASYNC_RECOVERY: 1h: DISABLE_ASYNC_RECOVERY_YES (default)
9	R/W	1	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_USE_BIST_F_MODE: 1h: USE_BIST_F_MODE_YES (default)
8	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_F: 0h: F_DEFAULT (default)
7	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_BIST_DWORD: 0h: BIST_DWORD_DEFAULT (default)
6	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_BIST_L_RDY: 0h: BIST_L_RDY_DEFAULT (default)
5	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_BIST_T_RDY: 0h: BIST_T_RDY_DEFAULT (default)
4	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_P: 0h: P_DEFAULT (default)
3	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_S: 0h: S_DEFAULT (default)
2	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_T: 0h: T_DEFAULT (default)
1	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_A: 0h: A_DEFAULT (default)
0	R/W	0	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL_CLR: 0h: CLR_DEFAULT (default)

This register has the Dword that is used in the second and third Dword of BIST FIS.

This register applies to SATA0 register space only.

Offset: 0x31c | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA0_AHCI_HBA_BIST_DWORD_DATA: 0h: DATA_DEFAULT (default)

T_SATA0_AHCI_HBA_SPARE_1

This register applies to SATA0 register space only.

Offset: 0x320 | Read/Write: R/W | Reset: 0x00030D40

Bit	R/W	Reset	Description
31:25	R/W	0	T_SATA0_AHCI_HBA_SPARE_1_RSVD: NOTE: this field is reset by Cold Reset 0h: RSVD_ZERO (default)
24	R	0	Reserved
23:0	R/W	030D40h	T_SATA0_AHCI_HBA_SPARE_1_MS_TIMER_CNT: This gives the count of fpci_clk for counting to millisecond. NOTE: this field is reset by Cold Reset 30D40h: MS_TIMER_CNT_DEFAULT (default) 30D40h: MS_TIMER_CNT_200MHZ 3D090h: MS_TIMER_CNT_250MHZ 493E0h: MS_TIMER_CNT_300MHZ 55730h: MS_TIMER_CNT_350MHZ 61A80h: MS_TIMER_CNT_400MHZ 7A120h: MS_TIMER_CNT_500MHZ

T_SATA0_AHCI_HBA_SPARE_2

This register applies to SATA0 register space only.

Offset: 0x324 | Read/Write: R/W | Reset: 0x00009000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_RSVD: 0h: RSVD_ZERO (default)
15	R/W	1	T_SATA0_AHCI_HBA_SPARE_2_LINK_IN_PROCESS_OF_LOW_POWER: 1h: LINK_IN_PROCESS_OF_LOW_POWER_DEFAULT (default)
14	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_DONT_USE_LD_IN_COUNTER: 0h: DONT_USE_LD_IN_COUNTER_INIT (default)
13	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_SW_COMWAKE_CONTINUOUS: 1h: SW_COMWAKE_CONTINUOUS_ENABLE 0h: SW_COMWAKE_CONTINUOUS_DISABLE (default)
12	R/W	1	T_SATA0_AHCI_HBA_SPARE_2_ASYNC_RECOVERY_ENABLE: 1h: ASYNC_RECOVERY_ENABLE_TURE (default) 0h: ASYNC_RECOVERY_ENABLE_FALSE
11:8	R/W	0	T_SATA0_AHCI_HBA_SPARE_2_ASYNC_RECOVERY_TIMER_SEL: 0h: ASYNC_RECOVERY_TIMER_SEL_DEFAULT (default)
7:0	R	0	Reserved

T_SATA0_AHCI_HBA_DYN_CLK_CLAMP

DYN_CLK_CLAMP_TIMER is used to wait for some time after the transactions on FPCI and CSM are done to shut down the devclk.

This register applies to SATA0 register space only.

Offset: 0x328 | Read/Write: R/W | Reset: 0x02000207

Bit	R/W	Reset	Description
31:24	R/W	02h	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_TIMER_PSM: 2h: TIMER_PSM_INIT (default)

Bit	R/W	Reset	Description
23:16	R/W	0	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_SPARE_1: 0h: SPARE_1_INIT (default)
15:8	R/W	02h	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_TIMER: 2h: TIMER_DEFAULT (default)
7:3	R/W	0	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_SPARE_0: 0h: SPARE_0_INIT (default)
2:0	R/W	7h	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP_SPARE_0_2: 7h: SPARE_0_2_INIT (default)

T_SATA0_AHCI_CFG_ERR_CTRL

This register implement settings for controlling behavior of AHCI SM during error conditions.

This register applies to SATA0 register space only.

Offset: 0x32c | Read/Write: R/W | Reset: 0x0000003F

Bit	R/W	Reset	Description
31:6	R	0	Reserved
5	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_ENABLE_PXIS_IFS_DATA_FIS_CRC_ERROR: 1h: ENABLE_PXIS_IFS_DATA_FIS_CRC_ERROR_INIT (default)
4	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_ENABLE_PXIS_IFS_D2H_SYNC_ESCAPE: 1h: ENABLE_PXIS_IFS_D2H_SYNC_ESCAPE_INIT (default)
3	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_TX_BUFFER_ON_FATAL_ERROR: 1h: CLR_TX_BUFFER_ON_FATAL_ERROR_INIT (default)
2	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_T2P_FIFO_ON_FATAL_ERROR: 1h: CLR_T2P_FIFO_ON_FATAL_ERROR_INIT (default)
1	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_P2T_FIFO_ON_FATAL_ERROR: 1h: CLR_P2T_FIFO_ON_FATAL_ERROR_INIT (default)
0	R/W	1	T_SATA0_AHCI_CFG_ERR_CTRL_CLR_CMD_FIFO_ON_FATAL_ERROR: 1h: CLR_CMD_FIFO_ON_FATAL_ERROR_INIT (default)

T_SATA0_AHCI_HBA_CAP2_BKDR

This register has bit by bit backdoor fields for CAP2 register according to the AHCI 1.3 specification.

This register applies to SATA0 register space only.

Offset: 0x330 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:6	R	0h	T_SATA0_AHCI_HBA_CAP2_BKDR_RSVD_31_6: 0h: RSVD_31_6_ZERO
3	R/W	1	T_SATA0_AHCI_HBA_CAP2_BKDR_SUPPORTS_DEVSLP: NOTE: this field is reset by Cold Reset 1h: SUPPORTS_DEVSLP_TRUE (default) 0h: SUPPORTS_DEVSLP_FALSE
2	R/W	1	T_SATA0_AHCI_HBA_CAP2_BKDR_AUTO_PARTIAL_TO_SLUMBER: NOTE: this field is reset by Cold Reset 1h: AUTO_PARTIAL_TO_SLUMBER_TRUE (default) 0h: AUTO_PARTIAL_TO_SLUMBER_FALSE
1	R	0h	T_SATA0>_AHCI_HBA_CAP2_BKDR_RSVD_1: 0h: RSVD__ZERO
0	R/W	0	T_SATA0_AHCI_HBA_CAP2_BKDR_BOH: NOTE: this field is reset by Cold Reset 1h: BOH_TRUE 0h: BOH_FALSE (default)

T_SATA0_AHCI_HBA_CTL_1

This register applies to SATA0 register space only.

Offset: 0x338 | Read/Write: R/W | Reset: 0xC12994C0

Bit	R/W	Reset	Description
31	R/W	1	T_SATA0_AHCI_HBA_CTL_1_UNCLAMP_HBA_CTRLR: 1h: UNCLAMP_HBA_CTRLR_INIT (default)
30	R/W	1	T_SATA0_AHCI_HBA_CTL_1_INVALIDATE_PRDS_AT_FETCH: 1h: INVALIDATE_PRDS_AT_FETCH_INIT (default)
29	R/W	0	T_SATA0_AHCI_HBA_CTL_1_PREFETCH_TX_DATA_NO_DELAY: 0h: PREFETCH_TX_DATA_NO_DELAY_INIT (default)
28	R/W	0	T_SATA0_AHCI_HBA_CTL_1_CHK_L_IDLE_IN_WAKE_LINK: 0h: CHK_L_IDLE_IN_WAKE_LINK_INIT (default)
27	R/W	0	T_SATA0_AHCI_HBA_CTL_1_CHK_L_IDLE_IN_LOW_POWER: 0h: CHK_L_IDLE_IN_LOW_POWER_INIT (default)
26	R/W	0	T_SATA0_AHCI_HBA_CTL_1_SKIP_SCTL_DET_WR_COMRESET: NOTE: this field is reset by Cold Reset 0h: SKIP_SCTL_DET_WR_COMRESET_INIT (default)
25	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ALWAYS_DENY_PMREQ: 0h: ALWAYS_DENY_PMREQ_INIT (default)
24	R/W	1	T_SATA0_AHCI_HBA_CTL_1_SKIP_SCTL_DET_WR_OFFLINE: NOTE: this field is reset by Cold Reset 1h: SKIP_SCTL_DET_WR_OFFLINE_INIT (default)
23:22	R/W	0	T_SATA0_AHCI_HBA_CTL_1_FETCH_REQ_TAKEN: 0h: FETCH_REQ_TAKEN_INIT (default)
21	R/W	1	T_SATA0_AHCI_HBA_CTL_1_NON_NCQ_IFS_TEMP_OFF: 1h: NON_NCQ_IFS_TEMP_OFF_INIT (default)
20	R/W	0	T_SATA0_AHCI_HBA_CTL_1_BM_HOLD_ACTV_TILL_INTR: 0h: BM_HOLD_ACTV_TILL_INTR_INIT (default)
19	R/W	1	T_SATA0_AHCI_HBA_CTL_1_PRD_SM_RXDATA_PREMATURE_END: 1h: PRD_SM_RXDATA_PREMATURE_END_INIT (default)
18	R/W	0	T_SATA0_AHCI_HBA_CTL_1_GOTO_PM_AGGR_FROM_P_IDLE: 1h: GOTO_PM_AGGR_FROM_P_IDLE_YES 0h: GOTO_PM_AGGR_FROM_P_IDLE_NO (default)
17	R/W	0	T_SATA0_AHCI_HBA_CTL_1_SEND_DATA_HDR_EARLY: 0h: SEND_DATA_HDR_EARLY_INIT (default)
16	R/W	1	T_SATA0_AHCI_HBA_CTL_1_FIS_SM_USE_RISEDGE_EOF: 1h: FIS_SM_USE_RISEDGE_EOF_INIT (default)
15	R/W	1	T_SATA0_AHCI_HBA_CTL_1_DETECT_B2B_NCQ_TAG_REPEAT: 1h: DETECT_B2B_NCQ_TAG_REPEAT_INIT (default)
14	R/W	0	T_SATA0_AHCI_HBA_CTL_1_NO_PRDBC_UPDATE: 0h: NO_PRDBC_UPDATE_INIT (default)
13:12	R/W	1	T_SATA0_AHCI_HBA_CTL_1_ARB_PARK_MODE: 1h: ARB_PARK_MODE_LAST_GNTED (default) 0h: ARB_PARK_MODE_PORT0
11	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ARB_GNT_CFG: 0h: ARB_GNT_CFG_INIT (default)
10	R/W	1	T_SATA0_AHCI_HBA_CTL_1_WAR_NDR_ACCEPT_ARC1_ISSUE: 1h: WAR_NDR_ACCEPT_ARC1_ISSUE_INIT (default)
9	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_CMD_PREFETCH_2: 0h: DISABLE_BKGD_CMD_PREFETCH_2_INIT (default)
8	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ISSUE_ORDER_CMD_SEL_IN_CBS: 0h: ISSUE_ORDER_CMD_SEL_IN_CBS_INIT (default)
7	R/W	1	T_SATA0_AHCI_HBA_CTL_1_INTR_PENDING_ON_CH_SEL: 1h: INTR_PENDING_ON_CH_SEL_INIT (default)
6	R/W	1	T_SATA0_AHCI_HBA_CTL_1_READ_OUT_BM_START: 1h: READ_OUT_BM_START_INIT (default)
5	R/W	0	T_SATA0_AHCI_HBA_CTL_1_SYNC_ESC_IN_ALPM: 0h: SYNC_ESC_IN_ALPM_INIT (default)
4	R/W	0	T_SATA0_AHCI_HBA_CTL_1_ENABLE_PRD_SPLIT_IN_CBS: 0h: ENABLE_PRD_SPLIT_IN_CBS_INIT (default)

Bit	R/W	Reset	Description
3	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_ACMD_PREFETCH: 0h: DISABLE_BKGD_ACMD_PREFETCH_INIT (default)
2	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_DATA_PREFETCH: 0h: DISABLE_BKGD_DATA_PREFETCH_INIT (default)
1	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_PRD_PREFETCH: 0h: DISABLE_BKGD_PRD_PREFETCH_INIT (default)
0	R/W	0	T_SATA0_AHCI_HBA_CTL_1_DISABLE_BKGD_CMD_PREFETCH_1: 0h: DISABLE_BKGD_CMD_PREFETCH_1_INIT (default)

T_SATA0_AHCI_HBA_PI_BKDR

This register applies to SATA0 register space only.

Offset: 0x33c | Read/Write: R/W | Reset: 0xXXXXXX01

Bit	R/W	Reset	Description
31:8	R	0h	T_SATA0_AHCI_HBA_PI_BKDR_31_8_RSVD: 0h: 31_8_RSVD_ZERO
7:0	R/W	1	T_SATA0_AHCI_HBA_PI_BKDR_PORTS_IMPL: NOTE: this field is reset by Cold Reset 0h: PORTS_IMPL_ZERO 3h: PORTS_IMPL_TWO Fh: PORTS_IMPL_FOUR 1h: PORTS_IMPL_DEFAULT (default)

T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL

This register applies to SATA0 register space only.

Offset: 0x340 | Read/Write: R/W | Reset: 0xAA0006D4

Bit	R/W	Reset	Description
31:24	R/W	AAh	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_SPARE: AAh: SPARE_INIT (default)
23:20	R	0	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_ENGAGED: 0h: ENGAGED_INIT (default)
19:18	R	0	Reserved
17:12	R/W	0	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_DMA_XFER_CNT: 0h: DMA_XFER_CNT_INIT (default)
11:8	R/W	6h	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_FIFO_LEVEL: 6h: FIFO_LEVEL_INIT (default)
7	R/W	1	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_ENABLE: 1h: ENABLE_INIT (default)
6	R/W	1	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_ATAPI_EN: 1h: ATAPI_EN_INIT (default)
5:0	R/W	14h	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL_PRDBC: 14h: PRDBC_INIT (default)

T_SATA<0>_CFG

Offset: 0x370 | Read/Write: R/W | Reset: 0x0F000000 (SATA) / 0x0F3XXXXX (SATA0)

Bit	R/W	Reset	Description
31:24	R/W	0Fh	T_SATA<0>_CFG_CTRL_TICKS_FOR_MASTER_TO: This is used in master slave emulation. Master waits for TICKS_FOR_MASTER_TO * 10**8 fpci_clk_sata cycles before timing out while waiting for response from slave drive on a device diagnostic command. Fh: CTRL_TICKS_FOR_MASTER_TO_250MHZ Fh: CTRL_TICKS_FOR_MASTER_TO_INIT (default)
23:0	0	R	Reserved (SATA only)

Bit	R/W	Reset	Description
23	R/W	0	T_SATA0_CFG_WR_ALLOWED_SHAD_REG: This bit when set will allow writes to shadow register even when BSY/ DRQ is one 0h: WR_ALLOWED_SHAD_REG_INIT (default)
22	R/W	0	T_SATA0_CFG_HOT_RESET_ON_BM_START_ALONE: 0h: HOT_RESET_ON_BM_START_ALONE_INIT (default)
21	R/W	1	T_SATA0_CFG_HOT_RESET_OLD_BEHAVIOUR: 1h: HOT_RESET_OLD_BEHAVIOUR_INIT (default)
20	R/W	1	T_SATA0_CFG_RESET_SREGS_ON_OOB_COMRESET: NOTE: this field is reset by Cold Reset 1h: RESET_SREGS_ON_OOB_COMRESET_TRUE (default) 0h: RESET_SREGS_ON_OOB_COMRESET_FALSE
19:0	R	0h	T_SATA0_CFG_RSVD_1: 0h: RSVD_1_VAL

T_SATA0_CTL_SHADOW

This register applies to SATA0 register space only.

Offset: 0x378 | Read/Write: R/W | Reset: 0x3XXXXXXX

Bit	R/W	Reset	Description
31	R/W	0	T_SATA0_CTL_SHADOW_PIO_DATA_READ_RETRY: 0h: PIO_DATA_READ_RETRY_NEW (default) 1h: PIO_DATA_READ_RETRY_OLD
30	R/W	0	T_SATA0_CTL_SHADOW_ALLOW_CTRL_WR_WHEN_DRIVE_NOT_PRESENT: 1h: ALLOW_CTRL_WR_WHEN_DRIVE_NOT_PRESENT_YES 0h: ALLOW_CTRL_WR_WHEN_DRIVE_NOT_PRESENT_NO (default)
29	R/W	1	T_SATA0_CTL_SHADOW_RET_ERROR_7F_WHEN_MASTER_ABSENT: 1h: RET_ERROR_7F_WHEN_MASTER_ABSENT_TRUE (default) 0h: RET_ERROR_7F_WHEN_MASTER_ABSENT_NO
28	R/W	1	T_SATA0_CTL_SHADOW_FIX_SRST_WHEN_ONE_DEVICE_NOT_PRESENT: 1h: FIX_SRST_WHEN_ONE_DEVICE_NOT_PRESENT_INIT (default)
27	R/W	0	T_SATA0_CTL_SHADOW_USE_ELONGATED_FIFO_HOT_RESET: 0h: USE_ELONGATED_FIFO_HOT_RESET_INIT (default)
26:0	R	0h	T_SATA0_CTL_SHADOW_RSVD: 0h: RSVD_VAL

T_SATA0_CFG_FPCI_0

This register applies to SATA0 register space only.

Offset: 0x430 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31	R/W	0	T_SATA0_CFG_FPCI_0_AHCI_MODE_DEP_ON_EN: When class code is 0106, ahci_mode will be one When class code is 0101, ahci_mode will be zero in other cases ahci_mode will depend on ahci_en if this bit is set, if bit is zero ahci_mode will be one NOTE: this field is reset by Cold Reset 1h: AHCI_MODE_DEP_ON_EN_YES 0h: AHCI_MODE_DEP_ON_EN_NO (default)
30	R/W	1	T_SATA0_CFG_FPCI_0_SEND_MSG_ON_NEW_INTR: This is used in the case of multiple MSI. This is to disbale/ enable sending interrupts when a new bit is written in pxis and if a some bits are cleared but not all and there is a posedge on a bit in pxie (see the AHCI specification, section 10.7.2) 1h: SEND_MSG_ON_NEW_INTR_INIT (default)
29	R/W	1	T_SATA0_CFG_FPCI_0_SEND_NEW_MSG_IS_NOT_ZERO: This is used in the case of a single MSI to send new message (intr), if software clears some bits of IS but not all (see the AHCI specification, section 10.7.2) 1h: SEND_NEW_MSG_IS_NOT_ZERO_INIT (default)
28:0	R	0h	T_SATA0_CFG_FPCI_0_RSVD: 0h: RSVD_VAL

T_SATA<0>_IDE1

When 40-bit addressing mode is enabled, bits [39:32] of the prd_base are loaded.

Offset: 0x490 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_IDE1_OLD_IDE_TIMING: 0h: OLD_IDE_TIMING_DEFAULT (default)

T_SATA0_CFG_ESATA_CTRL

This register implements settings for controlling behavior of the AHCI SM during error conditions.

This register applies to SATA0 register space only.

Offset: 0x494 | Read/Write: R/W | Reset: 0xXXXX0000

Bit	R/W	Reset	Description
31:16	R	0h	T_SATA0_CFG_ESATA_CTRL_RSVD: 0h: RSVD_VAL
15:8	R/W	0	T_SATA0_CFG_ESATA_CTRL_TX_PEAK_LIMIT: NOTE: this field is reset by Cold Reset 0h: TX_PEAK_LIMIT_INIT (default)
7:0	R/W	0	T_SATA0_CFG_ESATA_CTRL_TX_AMP_LIMIT: NOTE: this field is reset by Cold Reset 0h: TX_AMP_LIMIT_INIT (default)

T_SATA<0>_FEATURE

Feature Register

The feature options are shown in the following table.

Bit	Feature
31	NBP enable
30:13	Reserved
12:11	RAID function by SW programming
10	Reserved
9	Only 4 SATA ports supported
8	Software feature
7	Only 2 SATA ports supported
6	AHCI disabled
5	eSATA disabled
4	Aggressive power management disabled
3	Staggered spin-up disabled
2	iSCSI disabled
1	SATA Gen2 enabled
0	Advanced storage features disabled

Offset: 0x498 | Read/Write: R/W | Reset: 0xXX00XXXX

Bit	R/W	Reset	Description
31:24	R	0h	T_SATA<0>_FEATURE_RSVD_31_24: 0h: RSVD_31_24_VAL
23:18	R	0	T_SATA<0>_FEATURE_AHCI_ESATA_DIS: 3Fh: AHCI_ESATA_DIS_YES 0h: AHCI_ESATA_DIS_NO (default)
17:14	R	0	Reserved

Bit	R/W	Reset	Description
13	R	0h	T_SATA<0>_FEATURE_RSVD_13: 0h: RSVD_13_VAL
12:10	R	None	T_SATA<0>_FEATURE_RAID_FUNCTION_DIS:
9	R	0	T_SATA<0>_FEATURE_PORTS_2_3_DIS: 1h: PORTS_2_3_DIS_YES 0h: PORTS_2_3_DIS_NO (default)
8	R	0	T_SATA<0>_FEATURE_PORTS_0_1_DIS: 1h: PORTS_0_1_DIS_YES 0h: PORTS_0_1_DIS_NO (default)
7	R	0	Reserved
6	R	0h	T_SATA<0>_FEATURE_RSVD_6: 0h: RSVD_6_VAL 0h: RSVD_7_VAL
5	R	0h	T_SATA<0>_FEATURE_RSVD_5: 0h: RSVD_5_VAL
4	R	0	T_SATA<0>_FEATURE_AHCI_POWER_DIS: 1h: AHCI_POWER_DIS_YES 0h: AHCI_POWER_DIS_NO (default)
3	R	0	T_SATA<0>_FEATURE_FBS_DIS: 1h: FBS_DIS_YES 0h: FBS_DIS_NO (default)
2	R	0	T_SATA<0>_FEATURE_GEN3_EN: 1h: GEN3_EN_YES 0h: GEN3_EN_NO (default)
1	R	1	T_SATA<0>_FEATURE_GEN2_EN: 1h: GEN2_EN_YES (default) 0h: GEN2_EN_NO
0	R	0h	T_SATA<0>_FEATURE_RSVD_0: 0h: RSVD_0_VAL

T_SATA<0>_CTL1

Miscellaneous CTL1

Offset: 0x4a0 | Read/Write: R/W | Reset: 0xXXXXXX01

Bit	R/W	Reset	Description
31	R	0	Reserved
30:10	R	0h	T_SATA<0>_CTL1_RSVD_30_10: 0h: RSVD_30_10_VAL
9	R/W	1	T_SATA<0>_CTL1_BLK_NONPIO_IDP: BLK_NONPIO_IDP is used to suppress the nonpio read in the case of idp read. At default 1 value the second read will be blocked. At 0 both reads will occur. 1h: BLK_NONPIO_IDP_DEFAULT (default)
8:1	R	0	Reserved
0	R/W	1	T_SATA<0>_CTL1_SATA_ADNVCD_SPD_NEGO: ADNVCD_SPD_NEGO turns on an advanced way of detecting SATA speeds. SATA will mix detecting the two supported speeds. Otherwise it detects gen2 first and then gen1. NOTE: this field is reset by Cold Reset 1h: SATA_ADNVCD_SPD_NEGO_YES (default) 0h: SATA_ADNVCD_SPD_NEGO_NO

T_SATA<0>_BKDOOR_CC

This register is used to backdoor update the programming interface field and class code of the register at address 0x8 (T_SATA<0>_CFG_2[15:8]). Note that the default value of the programming interface field is 0x85 for SATA<0> and 0x8A for SATA1.

To update the programming interface field, the BIOS needs to do the following:

1. Read T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN - Address 54C

2. Write 1 to T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN
3. Write the desired programming interface value to T_SATA<0>_BKDOOR_CC[15:8]
4. Restore T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN to the value read in step#1

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x01018AXX (SATA) / 0x010185XX (SATA0)

Bit	R/W	Reset	Description
31:16	R/W	0101h	T_SATA<0>_BKDOOR_CC_CLASS_CODE: NOTE: this field is reset by Cold Reset 101h: CLASS_CODE_INIT (default)
15:8	R/W	8A (SATA) 85h (SATA0)	T_SATA<0>_BKDOOR_CC_PROG_IF: NOTE: this field is reset by Cold Reset 85h: PROG_IF_INIT (default)
7:0	R	0h	T_SATA<0>_BKDOOR_CC_RSVD_0: 0h: RSVD_0_VAL

T_SATA<0>_CFG_CTRL_1

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:2	R	0	Reserved
1	R/W	0	T_SATA<0>_CFG_CTRL_1_SATA_CAP: This bit controls whether or not SATA capability is enabled or not. By default it is enabled. NOTE: this field is reset by Cold Reset 1h: SATA_CAP_DISABLE 0h: SATA_CAP_ENABLE (default)
0	R/W	0	T_SATA<0>_CFG_CTRL_1_48BIT_ADDRESS_DISABLE: 48BIT_ADDRESS_DISABLE is used to disable 48-bit addressing mode. The SATA implementation of 48-bit addressing is such that this bit is no longer needed. It can be removed safely and is currently ignored in SATA. NOTE: this field is reset by Cold Reset 0h: 48BIT_ADDRESS_DISABLE_NO (default) 1h: 48BIT_ADDRESS_DISABLE_YES

T_SATA0_CFG_POWER_GATE

This register applies to SATA0 register space only.

Offset: 0x4ac | Read/Write: R/W | Reset: 0xXX000000

Bit	R/W	Reset	Description
31:24	R	0h	T_SATA0_CFG_POWER_GATE_RSVD: 0h: RSVD_VAL
23	R/W	0	T_SATA0_CFG_POWER_GATE_SSTS_RESTORED: 0h: SSTS_RESTORED_DEFAULT (default) 1h: SSTS_RESTORED_YES
22	R	0	T_SATA0_CFG_POWER_GATE_PHY_NOT_IN_SLUMBER: 0h: PHY_NOT_IN_SLUMBER_INIT (default)
21	R/W	0	T_SATA0_CFG_POWER_GATE_STOP_INTERRUPTS: 0h: STOP_INTERRUPTS_INIT (default)
20	R	0	T_SATA0_CFG_POWER_GATE_COUNT_FOR_PADS_DONE: 0h: COUNT_FOR_PADS_DONE_INIT (default)
19	R/W	0	T_SATA0_CFG_POWER_GATE_PADS_POWER_DOWN: 0h: PADS_POWER_DOWN_INIT (default)
18	R/W	0	T_SATA0_CFG_POWER_GATE_HW_WAKEUP_SUPPORT: 1h: HW_WAKEUP_SUPPORT_YES 0h: HW_WAKEUP_SUPPORT_NO (default)
17:2	R	0	T_SATA0_CFG_POWER_GATE_SM2SATA_PG_INFO: Indicates the data stored in SM unit - for now it has nothing 0h: SM2SATA_PG_INFO_DEFAULT (default)

1	R/W	0	T_SATA0_CFG_POWER_GATE_POWER_UNGATE_COMP: SW writes this bit to one when SW has restored all the registers necessary HW will write it back to zero 1h: POWER_UNGATE_COMP_YES 0h: POWER_UNGATE_COMP_NO (default) 1h: POWER_UNGATE_COMP_SET
0	R/W	0	T_SATA0_CFG_POWER_GATE_ENTER_PG: SW writes this bit to one when it wants the controller to enter Power Gating HW will write it back to zero 1h: ENTER_PG_YES 0h: ENTER_PG_NO (default) 1h: ENTER_PG_SET

T_SATA0_CFG_CTL_GLUE

This register applies to SATA0 register space only.

Offset: 0x4b0 | Read/Write: R/W | Reset: 0xXXXXXXCB

Bit	R/W	Reset	Description
31:9	R	0h	T_SATA0_CFG_CTL_GLUE_RSVD: 0h: RSVD_VAL
8	R/W	0	T_SATA_CFG_CTL_GLUE_ODD_DWORD_ALIGNMENT_FIX: 0h: ODD_DWORD_ALIGNMENT_FIX_INIT
6	R/W	1	T_SATA0_CFG_CTL_GLUE_RESET_COMMA_DETECTION_CNTR: NOTE: this field is reset by Cold Reset 1h: RESET_COMMA_DETECTION_CNTR_INIT (default)
5	R/W	0	T_SATA0_CFG_CTL_GLUE_HOLD_REQ_DEVCLK_CLAMP_2: NOTE: this field is reset by Cold Reset 0h: HOLD_REQ_DEVCLK_CLAMP_2_INIT (default)
4	R/W	0	T_SATA0_CFG_CTL_GLUE_HOLD_REQ_DEVCLK_CLAMP_1: NOTE: this field is reset by Cold Reset 0h: HOLD_REQ_DEVCLK_CLAMP_1_INIT (default)
3	R/W	1	T_SATA0_CFG_CTL_GLUE_LOCKDET_OVR_LOGIC_FIX: NOTE: this field is reset by Cold Reset 1h: LOCKDET_OVR_LOGIC_FIX_INIT (default)
2	R/W	0	T_SATA0_CFG_CTL_GLUE_OVERRIDE_LOCKDET_FOR_NVA: NOTE: this field is reset by Cold Reset 0h: OVERRIDE_LOCKDET_FOR_NVA_INIT (default)
1	R/W	1	T_SATA0_CFG_CTL_GLUE_USE_OVERRIDE_LOCKDET: NOTE: this field is reset by Cold Reset 1h: USE_OVERRIDE_LOCKDET_YES (default) 0h: USE_OVERRIDE_LOCKDET_NO
0	R/W	1	T_SATA0_CFG_CTL_GLUE_FIX_FPCI_WRDAT: When set to one adds a clock delay which is necessary for correct operation This bit will be removed once the platform emulation could do writes to BAR5 space NOTE: this field is reset by Cold Reset 1h: FIX_FPCI_WRDAT_YES (default) 0h: FIX_FPCI_WRDAT_NO

T_SATA0_CFG_CTL_FA

This register applies to SATA0 register space only.

Offset: 0x4bc | Read/Write: R/W | Reset: 0x00000083

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:3	R/W	10h	T_SATA0_CFG_CTL_FA_CFG_EATER: 10h: CFG_EATER_DEFAULT (default)
2	R/W	0	T_SATA0_CFG_CTL_FA_USE_CFG_EATER: 1h: USE_CFG_EATER_YES 0h: USE_CFG_EATER_NO (default)

Bit	R/W	Reset	Description
1	R/W	1	T_SATA0_CFG_CTL_FA_EAT_CLK: 1h: EAT_CLK_YES (default) 0h: EAT_CLK_NO
0	R/W	1	T_SATA0_CFG_CTL_FA_FPCI_CLK_IS_128MHZ: 1h: FPCI_CLK_IS_128MHZ_YES (default) 0h: FPCI_CLK_IS_128MHZ_NO

T_SATA<0>_PERF0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x00000800

Bit	R/W	Reset	Description
31:29	R	0	Reserved
28	R/W	0	T_SATA<0>_PERF0_DONT_DELAY_ROK: Setting DONT_DELAY_ROK turns on an optimization feature that should help performance on DMA reads. Transport layer then completes the current transfer even when busmaster has not yet drained all of the Receive FIFO. Setting this bit to _TRUE should result in better performance and therefore it is the preferred mode of operation. This is not the default Reset value and it is expected that the BIOS or Driver will enable this feature. NOTE: this field is reset by Cold Reset 0h DONT_DELAY_ROK_FALSE (default) 1h: DONT_DELAY_ROK_TRUE
27	R	0	Reserved
26:25	R/W	0	T_SATA<0>_PERF0_SATA_PLL_POWERDOWN: NOTE: this field is reset by Cold Reset 1h: SATA_PLL_POWERDOWN_YES 0h SATA_PLL_POWERDOWN_NO (default)
24:12	R	0	Reserved
11:7	R/W	10h	T_SATA<0>_PERF0_MIN_FPCI_CLK_FREQ: This field set the minimum frequency at which the fpci_clk should run in increments of 1/16th. It provides SW to override value set by frequency adjuster unit. Note that the actual frequency level is maximum of all the controllers. NOTE: this field is reset by Cold Reset 10h MIN_FPCI_CLK_FREQ_16 (default)
6:0	R	0	Reserved

T_SATA<0>_PERF1

This register is used to configure some of the SATA implementation options.

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00005A6E

Bit	R/W	Reset	Description
31:24	R/W	0	T_SATA<0>_PERF1_MAX_WATER_MARK: The MAX_WATER_MARK field tracks the maximum occupancy of the two i2f_fifos. The purpose of this field is to give us guidance on the setting of HIGH_WATER_MARK field. Ideally, the HIGH_WATER_MARK field should be set so that the MAX_WATER_MARK field almost hit full. Note that MAX_WATER_MARK field is only 8 bits wide, so it can only track the lower 8 bits of the fifo level. The MSB value can be inferred from the MSB value of the 9 bit HIGH_WATER_MARK field. T_SATA<0>_CFG_39_RXDATA_IB_DELAY[0] enables resetting SSTATUS_DET and SSTATUS_SPD fields to 0 when channels enter into partial or slumber low power modes. By default this bit is 0 and these fields are not reset. NOTE: this field is reset by Cold Reset 0h MAX_WATER_MARK_0 (default)
23:16	R	0	Reserved
15:8	R/W	5Ah	T_SATA<0>_PERF1_HIGH_WATER_MARK_PIO: Bit 13:8 of the register is used for PIO reads high water mark. For PIO reads, fifo depth is 48. But each entry is 32bit, that is it contains only one dword. So, if mark is set at Y, there is headroom for (48-Y) dwords for drive to react to us sending HOLD. 5Ah HIGH_WATER_MARK_PIO_INIT (default)

Bit	R/W	Reset	Description
7:0	R/W	6Eh	T_SATA<0>_PERF1_HIGH_WATER_MARK: This field configures the read fifo limit at which point the host starts sending HOLD primitives to back-off the data transfer from device. This field is only relevant during READs and should typically be set to as high a value without running the risk of overflowing the Fifo. Bit5:0 of the register is used for DMA read high water mark. For DMA reads, fifo depth is 48. And, each entry is 64bit, that is it contains two dwords. So, if the watermark is set at X then there is headroom for (48-X)*2 dwords for drive to react to controller sending HOLD. 6Eh HIGH_WATER_MARK_INIT (default)

T_SATA0_SPARE_0

Spare register on cold reset with zero as the default value.

This register applies to SATA0 register space only.

Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA0_SPARE_0_RSVD: 0h RSVD_DEFAULT (default)

T_SATA0_SPARE_1

Spare register on warm reset with zero as the default value.

This register applies to SATA0 register space only.

Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA0_SPARE_1_RSVD: 0h RSVD_DEFAULT (default)

T_SATA0_SPARE_2

Spare register on cold reset with 1 as the default value.

This register applies to SATA0 register space only.

Offset: 0x528 | Read/Write: R/W | Reset: 0xFFFFFFFF

Bit	R/W	Reset	Description
31:7	R/W	1FFFFFFh	T_SATA0_SPARE_2_RSVD: 1FFFFFFh RSVD_DEFAULT (default)
6	R/W	1	T_SATA0_SPARE_2_SPLIT_D2H_REG_FIS: 1h SPLIT_D2H_REG_FIS_DEFAULT (default)
5	R/W	1	T_SATA0_SPARE_2_LATCH_PXCMD_FRE: 1h LATCH_PXCMD_FRE_DEFAULT (default)
4	R/W	1	T_SATA0_SPARE_2_SET_PXCMD_FR_ON_FRE: 1h SET_PXCMD_FR_ON_FRE_DEFAULT (default)
3	R/W	1	T_SATA0_SPARE_2_ERR_CLEANUP_WAIT_P_IDLE: 1h ERR_CLEANUP_WAIT_P_IDLE_DEFAULT (default)
2	R/W	1	T_SATA0_SPARE_2_FIS_PRO_WAIT_P_IDLE: 1h FIS_PRO_WAIT_P_IDLE_DEFAULT (default)
1	R/W	1	T_SATA0_SPARE_2_WAIT_DATA_DFIFO_POP: 1h WAIT_DATA_DFIFO_POP_DEFAULT (default)
0	R/W	1	T_SATA0_SPARE_2_REM_TWO_ALIGNS_IN_BIST_L: 1h REM_TWO_ALIGNS_IN_BIST_L_DEFAULT (default)

T_SATA0_SPARE_3

Spare register on warm reset with 1 as the default value.

This register applies to SATA0 register space only.

Offset: 0x52c | Read/Write: R/W | Reset: 0x0001A5E0

Bit	R/W	Reset	Description
31:19	R/W	0	T_SATA0_SPARE_3_RSVD: 0h RSVD_DEFAULT (default)
18:0	R/W	1A5E0h	T_SATA0_SPARE_3_COUNT_FOR_1_MS: 1A5E0h COUNT_FOR_1_MS_DEFAULT (default)

T_SATA<0>_CHXCFG1

Offset: 0x530 | Read/Write: R/W | Reset: 0xXXXX00XX

Bit	R/W	Reset	Description
31:16	R	0h	T_SATA<0>_CHXCFG1_RSVD_1: 0h RSVD_1_VAL
15:12	R/W	0	T_SATA<0>_CHXCFG1_PHY_CHX_RX_CTL: NOTE: this field is reset by Cold Reset 0h PHY_CHX_RX_CTL_INIT (default) 0h PHY_CHX_RX_CTL_VAL_0 1h PHY_CHX_RX_CTL_VAL_1 2h PHY_CHX_RX_CTL_VAL_2 3h PHY_CHX_RX_CTL_VAL_3 Fh PHY_CHX_RX_CTL_VAL_F
11:8	R/W	0	T_SATA<0>_CHXCFG1_PHY_CHX_TX_CTL: NOTE: this field is reset by Cold Reset 0h PHY_CHX_TX_CTL_INIT (default) 0h PHY_CHX_TX_CTL_VAL_0 1h PHY_CHX_TX_CTL_VAL_1 2h PHY_CHX_TX_CTL_VAL_2 3h PHY_CHX_TX_CTL_VAL_3 Fh PHY_CHX_TX_CTL_VAL_F
7:2	R	0h	T_SATA<0>_CHXCFG1_RSVD_0: 0h RSVD_0_VAL
1	R/W	0	T_SATA<0>_CHXCFG1_CHX_NEAR_LOOP_BACK: NOTE: this field is reset by Cold Reset 0h CHX_NEAR_LOOP_BACK_INACTIVE 1h CHX_NEAR_LOOP_BACK_ACTIVE 0h CHX_NEAR_LOOP_BACK_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHXCFG1_CHX_RESET: This can be used to reset the port in IDE mode of SATA controller. It is not advisable to use it in AHCI mode. NOTE: this field is reset by Cold Reset 0h CHX_RESET_INACTIVE 1h CHX_RESET_ACTIVE 0h CHX_RESET_DEFAULT (default)

T_SATA<0>_PHY_CTRL

PLL_SLEEP describes power setting of SATA2 internal PHY. Input to PHY is actually most active power setting from two SATA controllers. TX/RX power states are per port.

Offset: 0x534 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:5	R	0h	T_SATA<0>_PHY_CTRL_RSVD_1: 0h RSVD_1_VAL
4:3	R/W	0	T_SATA<0>_PHY_CTRL_PLL_SLEEP: NOTE: this field is reset by Cold Reset 0h PLL_SLEEP_INIT (default) 0h PLL_SLEEP_ACTIVE 1h PLL_SLEEP_3G_DISABLED 2h PLL_SLEEP_ALL_CLKS_DISABLED 3h PLL_SLEEP_PLL_DISABLED

Bit	R/W	Reset	Description
2:1	R	0h	T_SATA<0>_PHY_CTRL_RSVD_0: 0h RSVD_0_VAL
0	R/W	1	T_SATA<0>_PHY_CTRL_SLUMBER_DURING_D3: NOTE: this field is reset by Cold Reset 1h SLUMBER_DURING_D3_ENABLE (default) 0h SLUMBER_DURING_D3_DISABLE

T_SATA<0>_LDT

LDT Unit ID Register

The CFG_19 register contains the LDT Unit ID(s) for the Device. The Unit IDs are used by the device as a transaction ID on the LDT interface to uniquely identify where the request originated. Some devices will support 2 Unit IDs, 1 for isochronous requests and the other for non-isochronous requests.

Offset: 0x53c | Read/Write: R/W | Reset: 0x000000XX

Bit	R/W	Reset	Description
31:13	R	0	Reserved
12:8	R/W	0	T_SATA<0>_LDT_NON_ISO_UNIT_ID: Non-ISO Unit ID. 0h NON_ISO_UNIT_ID_0 (default) 9h NON_ISO_UNIT_ID_9 1h NON_ISO_UNIT_ID_1 2h NON_ISO_UNIT_ID_2 1Fh NON_ISO_UNIT_ID_31
7:5	R	0	Reserved
4:0	R	0h	T_SATA<0>_LDT_ISO_UNIT_ID: ISO Unit ID. Hard-wired to 0. 0h ISO_UNIT_ID_0 1h ISO_UNIT_ID_1 2h ISO_UNIT_ID_2 1Fh ISO_UNIT_ID_31

T_SATA<0>_CTRL

Offset: 0x540 | Read/Write: R/W | Reset: 0x60300007

Bit	R/W	Reset	Description
31	R	0	Reserved
30	R/W	1	T_SATA<0>_CTRL_ENABLE_CH2_CH4_INTR: NOTE: this field is reset by Cold Reset 1h ENABLE_CH2_CH4_INTR_YES (default) 0h ENABLE_CH2_CH4_INTR_NO
29	R/W	1	T_SATA<0>_CTRL_ENABLE_CH1_CH3_INTR: NOTE: this field is reset by Cold Reset 1h ENABLE_CH1_CH3_INTR_YES (default) 0h ENABLE_CH1_CH3_INTR_NO
28	R	0	Reserved
27	R/W	0	T_SATA<0>_CTRL_NP_RSP_ERROR_INTR: NOTE: this field is reset by Cold Reset 1h NP_RSP_ERROR_INTR_YES 0h NP_RSP_ERROR_INTR_NO (default)
26	R/W	0	T_SATA<0>_CTRL_PW_CMD_ERROR_INTR: NOTE: this field is reset by Cold Reset 1h PW_CMD_ERROR_INTR_YES 0h PW_CMD_ERROR_INTR_NO (default)
25	R/W	0	T_SATA<0>_CTRL_NP_RSP_ERROR_INTR_EN: NOTE: this field is reset by Cold Reset 1h NP_RSP_ERROR_INTR_EN_YES 0h NP_RSP_ERROR_INTR_EN_NO (default)

Bit	R/W	Reset	Description
24	R/W	0	T_SATA<0>_CTRL_PW_CMD_ERROR_INTR_EN: NOTE: this field is reset by Cold Reset 1h PW_CMD_ERROR_INTR_EN_YES 0h PW_CMD_ERROR_INTR_EN_NO (default)
23:22	R	0	Reserved
21	R/W	1	T_SATA<0>_CTRL_CH4_EN: NOTE: this field is reset by Cold Reset 1h CH4_EN_YES (default) 0h CH4_EN_NO
20	R/W	1	T_SATA<0>_CTRL_CH3_EN: NOTE: this field is reset by Cold Reset 1h CH3_EN_YES (default) 0h CH3_EN_NO
19:4	R	0	Reserved
3	R/W	0	T_SATA<0>_CTRL_INTF_ERR_HANDLING_EN: This is used to enable an advanced feature to handle SATA interface errors. It includes detecting premature CRC in non-AHCI mode and resetting transport layer on comreset. The interface errors are mostly seen on emulation systems. NOTE: this field is reset by Cold Reset 1h INTF_ERR_HANDLING_EN_YES 0h INTF_ERR_HANDLING_EN_NO (default)
2	R/W	1	T_SATA<0>_CTRL_BAR5_SPACE_EN: BAR5_SPACE_EN bit enables use of BAR5 space. So, if this bit is written with zero then SATA will not accept any cycles to BAR5 (starting from 1KB to 4KB). By default this bit is low. NOTE: this field is reset by Cold Reset 0h BAR5_SPACE_EN_NO 1h BAR5_SPACE_EN_YES (default)
1	R/W	1	T_SATA<0>_CTRL_PRI_CHANNEL_EN: Enables the primary SATA channel. Value 1 denotes that the respective channel is enabled and 0 indicates disabled channel. NOTE: this field is reset by Cold Reset 0h PRI_CHANNEL_EN_NO 1h PRI_CHANNEL_EN_YES (default)
0	R/W	1	T_SATA<0>_CTRL_SEC_CHANNEL_EN: Enables the secondary SATA channel. Value 1 denotes that the respective channel is enabled and 0 indicates disabled channel. NOTE: this field is reset by Cold Reset 0h SEC_CHANNEL_EN_NO 1h SEC_CHANNEL_EN_YES (default)

T_SATA<0>_CFG_SATA

Offset: 0x54c | Read/Write: R/W | Reset: 0xXXXXX02X

Bit	R/W	Reset	Description
31:17	R	0h	T_SATA<0>_CFG_SATA_RSVD_1: 0h RSVD_1_VAL
16	R/W	0 (SATA) 1 (SATA0)	T_SATA<0>_CFG_SATA_FORCE_NATIVE: NOTE: this field is reset by Cold Reset 1h FORCE_NATIVE_ENABLED (SATA0 default) 0h FORCE_NATIVE_DISABLED (SATA default)
15	R/W	0	T_SATA<0>_CFG_SATA_CTRL_ALT_UID_SCHEME: NOTE: this field is reset by Cold Reset 0h CTRL_ALT_UID_SCHEME_INIT (default)
14	R	Unknown	T_SATA<0>_CFG_SATA_CTRL_SATA_GEN3_CAPABLE: 1h CTRL_SATA_GEN3_CAPABLE_YES 0h CTRL_SATA_GEN3_CAPABLE_NO
13	R	Unknown	T_SATA<0>_CFG_SATA_CTRL_SATA_GEN2_CAPABLE: SATA_GEN2_CAPABLE is read-only register that reflects the value of bond_sata2. If high, GEN2 speed is enabled, if low, only GEN1 speed is available. This register is also used to send near end loopback and reset the channels. The MIN_ON_TIME bits set the additional time the activity led remains lit after a transaction. This keeps the LED from being a dim flicker during short burst. 1h CTRL_SATA_GEN2_CAPABLE_YES 0h CTRL_SATA_GEN2_CAPABLE_NO

Bit	R/W	Reset	Description
12	R/W	0	T_SATA<0>_CFG_SATA_BACKDOOR_PROG_IF_EN: NOTE: this field is reset by Cold Reset 1h BACKDOOR_PROG_IF_EN_YES 0h BACKDOOR_PROG_IF_EN_NO (default)
11:7	R/W	0	T_SATA<0>_CFG_SATA_PORT2UNITID_MAPPING: NOTE: this field is reset by Cold Reset 0h PORT2UNITID_MAPPING_DEFAULT (default)
6	R/W	0	T_SATA<0>_CFG_SATA_MSI_CAP_DISABLE: Normally, the T_SATA<0>_CFG_17_NEXT_PTR field points to T_SATA<0>_MSI_CTRL (0xB0). However, if SATA_MSI_CAP_DISABLE is set to _YES, then the T_SATA<0>_CFG_17_NEXT_PTR field would point to _NULL. If future capabilities are added beyond the MSI pointer, then care should be taken to ensure that CFG_17_NEXT_PTR bypasses the T_SATA<0>_MSI_CTRL address and points to the next capabilities pointer instead of _NULL. NOTE: this field is reset by Cold Reset 0h MSI_CAP_DISABLE_NO (default) 1h MSI_CAP_DISABLE_YES
5	R/W	1	T_SATA<0>_CFG_SATA_MSIX_CAP_DISABLE: 0h MSIX_CAP_DISABLE_NO 1h MSIX_CAP_DISABLE_YES (default)
4	R/W	0	T_SATA<0>_CFG_SATA_USE_40B_ADDR: If this bit is set then SATA uses NVIDIA style 40b addressing while DMAing data. It gets upper 8 bits from the 23:16 bits of the prd count associated with the prd address. NOTE: this field is reset by Cold Reset 0h USE_40B_ADDR_NO (default) 1h USE_40B_ADDR_YES
3	R/W	0	T_SATA<0>_CFG_SATA_ERROR_HANDLING: NOTE: this field is reset by Cold Reset 0h ERROR_HANDLING_NO (default) 1h ERROR_HANDLING_YES
2	R/W	0	T_SATA<0>_CFG_SATA_CTRL_RAID_MODE_CTRL: CTRL_RAID_MODE was originally intended to allow customers to downgrade their RAID capability in the SBIOS. Setting this bit to 0 would allow only RAID 0 and RAID 1 support in the driver while setting to 1 would allow all RAID support. Setting this bit to 1 modifies the DeviceID of the SATA controller which would be a key to the driver. NOTE: this field is reset by Cold Reset 0h CTRL_RAID_MODE_CTRL_OEM (default) 1h CTRL_RAID_MODE_CTRL_CHANNEL
1	R	Unknown	T_SATA<0>_CFG_SATA_CTRL_STORAGE_FEATURE: STORAGE_FEATURE goes to both SATA and IDE config spaces. This is a general purpose bit that SW can use to distinguish between Advanced and Basic storage controller modes. This bit will be interpreted to either allow or not allow RAID 0+1 or 5 mode. ADVANCED mode will allow all RAID modes and BASIC will allow only RAID 0 and RAID 1 modes. It is strongly encouraged to keep the meaning of this STORAGE_FEATURE bit the same for future chips and to create new bits if necessary. This will allow SW compatibility. 1h CTRL_STORAGE_FEATURE_BASIC 0h CTRL_STORAGE_FEATURE_ADVANCED
0	R	0h	T_SATA<0>_CFG_SATA_RSVD_0: 0h RSVD_0_VAL

T_SATA<0>_CFG_MISC

Offset: 0x550 | Read/Write: R/W | Reset: 0x00000163

Bit	R/W	Reset	Description
31:12	R	0	Reserved
11:8	R/W	1	T_SATA<0>_CFG_MISC_PHY_RESET_USAGE_MODE: Bit 0 controls use of lockdet as phy reset. Bit 1 when high lets us update ata status reg quickly during pio-writes. By default its high. Bit 2 controls pi_reset, it controls whether to use lockdet or a cfg bit to use as reset 1h INIT (default)
7:4	R/W	6h	T_SATA<0>_CFG_MISC_LINK_SM_MODE: Bit 0 enables holda bypass. Bits 3 and 2 control l_idle features. 6h INIT (default)

Bit	R/W	Reset	Description
3:0	R/W	3h	T_SATA<0>_CFG_MISC_PHY_OOB_SEQ_MODE: bit 0 controls whether we need to back off sending comwake when we receive comwake from drive Bit 1 is used to control our comreset behavior. If this bit is set to 1 then we send only one COMRESET to drive when scontrol_det[0] bit is written. If this bit is 0 then we keep sending COMRESET sequences as long as scontrol_det bit is active. 3h INIT (default)

T_SATA<0>_LOWPOWER_COUNT

Serial ATA Control Register

Offset: 0x554 | Read/Write: R/W | Reset: 0xXXXXXX44

Bit	R/W	Reset	Description
31:8	R	0h	T_SATA<0>_LOWPOWER_COUNT_RSVD: 0h RSVD_31_8
7:4	R/W	4h	T_SATA<0>_LOWPOWER_COUNT_SLUMBER: 4h SLUMBER_INIT (default)
3:0	R/W	4h	T_SATA<0>_LOWPOWER_COUNT_PARTIAL: 4h PARTIAL_INIT (default)

T_SATA<0>_AO_SPARE_0

Spare register on cold reset with zero as the default value

Offset: 0x580 | Read/Write: R/W | Reset: 0x7FFFFFFF

Bit	R/W	Reset	Description
31:0	R/W	7FFFFFFFh	T_SATA<0>_AO_SPARE_0_RSVD: 7FFFFFFFh RSVD_DEFAULT (default)

T_SATA<0>_AO_SPARE_1

Spare register on cold reset with zero as the default value

Offset: 0x584 | Read/Write: R/W | Reset: 0x7FFFFFFF

Bit	R/W	Reset	Description
31:0	R/W	7FFFFFFFh	T_SATA<0>_AO_SPARE_1_RSVD: 7FFFFFFFh RSVD_DEFAULT (default)

T_SATA<0>_INDEX

Index Mask Register

This register implements mask to select writes to channel specific registers. A write to any CHX field in this manual will affect corresponding register for channel n if the nth bit is set in this mask register. If mask register has no bits set then the write will not change any register.

A read from a CHX field will return data corresponding to the channel which has its mask set. If more than one bits are set in mask, then the returned data will be logical OR of register values corresponding to the channels which have mask bits set. If no mask bit is set then the returned value will be 0x00.

The following comment is no longer valid, separate registers are implemented for GEN[1,2,3] Further PEAK and AMP fields can be set differently for gen1 and gen2. A write to these fields will affect the setting for Genx if the bit corresponding to Genx is set in this register.

Offset: 0x680 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:4	R	0	Reserved

Bit	R/W	Reset	Description
3	R/W	0	T_SATA<0>_INDEX_CH4: 0h CH4_UNSELECTED (default) 1h CH4_SELECTED
2	R/W	0	T_SATA<0>_INDEX_CH3: 0h CH3_UNSELECTED (default) 1h CH3_SELECTED
1	R/W	0	T_SATA<0>_INDEX_CH2: 0h CH2_UNSELECTED (default) 1h CH2_SELECTED
0	R/W	0	T_SATA<0>_INDEX_CH1: 0h CH1_UNSELECTED (default) 1h CH1_SELECTED

T_SATA<0>_CHX_MISC

SATA Miscellaneous Control Register

Offset: 0x684 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:1	R	0	Reserved
0	R/W	0	T_SATA<0>_CHX_MISC_LED_DISABLE: 0h LED_DISABLE_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL1_GEN1

SATA PHY Control Register (GEN1)

Offset: 0x690 | Read/Write: R/W | Reset: 0x00000715

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29:26	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_DRV_CNTL: NOTE: this field is reset by Cold Reset 0h TX_DRV_CNTL_DEFAULT (default)
25:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_PEAK_PRE: NOTE: this field is reset by Cold Reset 0h TX_PEAK_PRE_DEFAULT (default)
19:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_CMADJ: NOTE: this field is reset by Cold Reset 0h TX_CMADJ_DEFAULT (default)
15:8	R/W	07h	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_PEAK: This field contains the PEAK values for GEN1. NOTE: this field is reset by Cold Reset 7h TX_PEAK_DEFAULT (default) 5h TX_PEAK_DEFAULT2 4h TX_PEAK_DEFAULT3 Ah TX_PEAK_DEFAULT4 Eh TX_PEAK_ESATA
7:0	R/W	15h	T_SATA<0>_CHX_PHY_CTRL1_GEN1_TX_AMP: This field contains the AMP values for GEN1 This register may have different values based on port usage (esata,gen1,gen) NOTE: this field is reset by Cold Reset 15h TX_AMP_DEFAULT (default) 12h TX_AMP_DEFAULT2 0Fh TX_AMP_DEFAULT3 18h TX_AMP_DEFAULT4 14h TX_AMP_ESATA

T_SATA<0>_CHX_PHY_CTRL1_GEN2

SATA PHY Control Register (GEN2)

Offset: 0x694 | Read/Write: R/W | Reset: 0x0000A018

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29:26	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_DRV_CNTL: 0h TX_DRV_CNTL_DEFAULT (default)
25:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_PEAK_PRE: 0h TX_PEAK_PRE_DEFAULT (default)
19:12	R/W	0Ah	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_PEAK: This field contains the PEAK values for gen1. Ah TX_PEAK_DEFAULT (default) Ah TX_PEAK_DEFAULT2 Ah TX_PEAK_DEFAULT3 Eh TX_PEAK_DEFAULT4 Eh TX_PEAK_ESATA
11:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_CMADJ: 0h TX_CMADJ_DEFAULT (default)
7:0	R/W	18h	T_SATA<0>_CHX_PHY_CTRL1_GEN2_TX_AMP: This field contains the AMP values for gen1 This register may have different values based on port usage (esata,gen1,gen) 18h TX_AMP_DEFAULT (default) 15h TX_AMP_DEFAULT2 12h TX_AMP_DEFAULT3 1Bh TX_AMP_DEFAULT4 1Ah TX_AMP_ESATA

T_SATA<0>_CHX_PHY_CTRL1_GEN3

SATA PHY Control Register (GEN3)

Offset: 0x698 | Read/Write: R/W | Reset: 0x0000E01F

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29:26	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_DRV_CNTL: NOTE: this field is reset by Cold Reset 0h TX_DRV_CNTL_DEFAULT (default)
25:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_PEAK_PRE: NOTE: this field is reset by Cold Reset 0h TX_PEAK_PRE_DEFAULT (default)
19:12	R/W	0Ah	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_PEAK: This field contains the PEAK values for gen1. NOTE: this field is reset by Cold Reset Ah TX_PEAK_DEFAULT (default) 14h TX_PEAK_ESATA
11:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_CMADJ: NOTE: this field is reset by Cold Reset 0h TX_CMADJ_DEFAULT (default)
7:0	R/W	1Fh	T_SATA<0>_CHX_PHY_CTRL1_GEN3_TX_AMP: This field contains the AMP values for gen1 This register may have different values based on port usage (esata,gen1,gen) NOTE: this field is reset by Cold Reset 1Fh TX_AMP_DEFAULT (default) 20h TX_AMP_ESATA

T_SATA<0>_CHX_PHY_CTRL2

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x69c | Read/Write: R/W | Reset: 0x01670137

Bit	R/W	Reset	Description
31:16	R	167h	T_SATA<0>_CHX_PHY_CTRL2_CDR_CNTL_GEN2: 167h: CDR_CNTL_GEN2_DEFAULT
15:0	R/W	137h	T_SATA<0>_CHX_PHY_CTRL2_CDR_CNTL_GEN1: 137h: CDR_CNTL_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL24

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6a0 | Read/Write: R/W | Reset: 0x000001C7

Bit	R/W	Reset	Description
15:0	R/W	1C7h	T_SATA<0>_CHX_PHY_CTRL24_CDR_CNTL_GEN3: 1C7h: CDR_CNTL_GEN3_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL25

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6a4 | Read/Write: R/W | Reset: 0x0000000

Bit	R/W	Reset	Description
31:0	R/W	0h	T_SATA<0>_CHX_PHY_CTRL25_DFE_CNTL_GEN2: 0h: DFE_CNTL_GEN2_DEFAULT

T_SATA<0>_CHX_PHY_CTRL26

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6a8 | Read/Write: R/W | Reset: 0x0000000

Bit	R/W	Reset	Description
31:0	R/W	0h	T_SATA<0>_CHX_PHY_CTRL26_DFE_CNTL_GEN3: 0h: DFE_CNTL_GEN3_DEFAULT

T_SATA<0>_CHX_PHY_CTRL3

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6b0 | Read/Write: R/W | Reset: 0x10004220

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29	R	0	T_SATA<0>_CHX_PHY_CTRL3_STATUS_RX_STAT_IDLE: 0h STATUS_RX_STAT_IDLE_DEFAULT (default)
28	R	1	T_SATA<0>_CHX_PHY_CTRL3_STATUS_TX_STAT_PRESENT: 1h STATUS_TX_STAT_PRESENT_DEFAULT (default)
27:26	R	0	T_SATA<0>_CHX_PHY_CTRL3_STATUS_RX_RATE: 0h STATUS_RX_RATE_DEFAULT (default) 0h STATUS_RX_RATE_GEN1 1h STATUS_RX_RATE_GEN2 2h STATUS_RX_RATE_GEN3

Bit	R/W	Reset	Description
25:24	R	0	T_SATA<0>_CHX_PHY_CTRL3_STATUS_TX_RATE: 0h STATUS_TX_RATE_DEFAULT (default) 0h STATUS_TX_RATE_GEN1 1h STATUS_TX_RATE_GEN2 2h STATUS_TX_RATE_GEN3
23	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_RATE_OVERRIDE: 0h RX_RATE_OVERRIDE_DISABLE (default) 1h RX_RATE_OVERRIDE_ENABLE
22	R	0	Reserved
21:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_RATE: 0h RX_RATE_GEN1 (default) 1h RX_RATE_GEN2 2h RX_RATE_GEN3
19	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_RATE_OVERRIDE: 0h TX_RATE_OVERRIDE_DISABLE (default) 1h TX_RATE_OVERRIDE_ENABLE
18	R	0	Reserved
17:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_RATE: 0h TX_RATE_GEN1 (default) 1h TX_RATE_GEN2 2h TX_RATE_GEN3
15	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_DATA_READY: 1h RX_DATA_READY_YES 0h RX_DATA_READY_NO (default)
14	R/W	1	T_SATA<0>_CHX_PHY_CTRL3_RX_DATA_EN: 1h RX_DATA_EN_DEFAULT (default)
13	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_DATA_EN_OVERRIDE: 1h TX_DATA_EN_OVERRIDE_YES 0h TX_DATA_EN_OVERRIDE_NO (default)
12	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_DATA_EN: TXDATA_EN and TXDATA_EN_OVERRIDE are used to enable/disable txd_en manually. PHY_CTRL3_TXD_EN and PHY_CTRL3_TXD_EN_OVERRIDE are used to enable/disable txd_en manually. 1h TX_DATA_EN_YES 0h TX_DATA_EN_NO (default)
11	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_RX_SLEEP_OVERRIDE: Enable a forced sleep mode on the SATA port, used for debug of the sleep modes. _NO = Normal Operation _YES = Forced sleep mode Note: The associated SLEEP_MODE register bits select the sleep mode forced. 1h RX_SLEEP_OVERRIDE_YES 0h RX_SLEEP_OVERRIDE_NO (default)
10	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_DATA_READY: 1h TX_DATA_READY_YES 0h TX_DATA_READY_NO (default)
9:8	R/W	2h	T_SATA<0>_CHX_PHY_CTRL3_RX_SLEEP: Select a sleep mode on the SATA port, used for debug of the sleep modes. 00 = Active 01 = Partial Slumber 10 = Slumber 11 = Disabled 2h RX_SLEEP_INIT (default) 0h RX_SLEEP_ACTIVE 1h RX_SLEEP_PARTIAL 2h RX_SLEEP_SLUMBER 3h RX_SLEEP_DISABLED
7	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_TX_SLEEP_OVERRIDE: Enable a forced sleep mode on the SATA port, used for debug of the sleep modes. _NO = Normal Operation _YES = Forced sleep mode Note: The associated SLEEP_MODE register bits select the sleep mode forced. 1h TX_SLEEP_OVERRIDE_YES 0h TX_SLEEP_OVERRIDE_NO (default)
6	R	0	Reserved

Bit	R/W	Reset	Description
5:4	R/W	2h	T_SATA<0>_CHX_PHY_CTRL3_TX_SLEEP: Select a sleep mode on the SATA port, used for debug of the sleep modes. 00 = Active 01 = Partial Slumber 10 = Slumber 11 = Disabled 2h TX_SLEEP_INIT (default) 0h TX_SLEEP_ACTIVE 1h TX_SLEEP_PARTIAL 2h TX_SLEEP_SLUMBER 3h TX_SLEEP_DISABLED
3	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_CKBUFPD_OVRD: 0h CKBUFPD_OVRD_DEFAULT (default)
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_CKBUFPD: 0h CKBUFPD_DEFAULT (default)
1	R	0	Reserved
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL3_IDDQ: 0h IDDQ_INIT (default)

T_SATA<0>_CHX_PHY_CTRL4

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6b4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:30	R	0	T_SATA<0>_CHX_PHY_CTRL4_SPARE_OUT: 0h SPARE_OUT_DEFAULT (default)
29:28	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_SPARE_IN: 0h SPARE_IN_DEFAULT (default)
27	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_PRBS_CHK_EN: 0h PRBS_CHK_EN_DEFAULT (default)
26	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TEST_EN: 0h TEST_EN_DEFAULT (default)
25	R	0	Reserved
24	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_RATE_MODE: 0h RATE_MODE_DEFAULT (default)
23:22	R	0	Reserved
21:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_RX_DIV: 0h RX_DIV_DEFAULT (default)
19:18	R	0	Reserved
17:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TX_DIV: 0h TX_DIV_DEFAULT (default)
15:14	R	0	Reserved
13	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_RX_CDR_RESET: 0h RX_CDR_RESET_DEFAULT (default)
12	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TX_SYNC: 0h TX_SYNC_IDLE (default) 1h TX_SYNC_NOW
11	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_FED_LOOP: 0h FED_LOOP_DEFAULT (default)
10:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_TX_DATA_MODE: 0h TX_DATA_MODE_NORMAL (default) 1h TX_DATA_MODE_PRBS_2_7 2h TX_DATA_MODE_0101010101 3h TX_DATA_MODE_1100110011 4h TX_DATA_MODE_0000011111 5h TX_DATA_MODE_0101111100

Bit	R/W	Reset	Description
7	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_FEA_LOOP: 0h FEA_LOOP_DEFAULT (default)
6:4	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_FEA_MODE: 0h FEA_MODE_DEFAULT (default)
3	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_NEA_LOOP: 0h NEA_LOOP_DEFAULT (default)
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_NED_LOOP: 0h NED_LOOP_DEFAULT (default)
1:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL4_NED_MODE: 0h NED_MODE_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL5

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6b8 | Read/Write: R/W | Reset: 0x00000208

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:24	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_CDR_MODE: 0h CDR_MODE_DEFAULT (default)
23:20	R	0	Reserved
19	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_TX_RDET: 0h TX_RDET_IDLE (default) 1h TX_RDET_START
18	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_RX_IDLE_MODE: 0h RX_IDLE_MODE_DEFAULT (default)
17	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_RX_IDLE_BYP: 0h RX_IDLE_BYP_DISABLE (default) 1h RX_IDLE_BYP_ENABLE
16	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_TX_RDET_BYP: 0h TX_RDET_BYP_DISABLE (default) 1h TX_RDET_BYP_ENABLE
15:14	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_RX_IDLE_T: 0h RX_IDLE_T_DEFAULT (default)
13:12	R/W	0	T_SATA<0>_CHX_PHY_CTRL5_TX_RDET_T: 0h TX_RDET_T_DEFAULT (default)
11:8	R/W	2h	T_SATA<0>_CHX_PHY_CTRL5_TX_SEL_LOAD: 2h TX_SEL_LOAD_DEFAULT (default)
7:4	R	0	Reserved
3:0	R/W	8h	T_SATA<0>_CHX_PHY_CTRL5_MISC_CNTL: 8h MISC_CNTL_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL6

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6bc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:29	R	0	Reserved
28:27	R	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_DATA: 0h: RX_BYP_DATA_DEFAULT
26	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_TERM_MODE: 0h RX_TERM_MODE_FALSE (default) 1h RX_TERM_MODE_TRUE

Bit	R/W	Reset	Description
25	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_TERM_EN: 0h RX_TERM_EN_FALSE (default) 1h RX_TERM_EN_TRUE
24	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_TERM_MODE: 0h TX_TERM_MODE_FALSE (default) 1h TX_TERM_MODE_TRUE
23:16	R	0	T_SATA<0>_CHX_PHY_CTRL6_MISC_OUT: 0h MISC_OUT_DEFAULT (default)
15	R	0	Reserved
14	R	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_IN: 0h RX_BYP_IN_DEFAULT (default)
13	R	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_IN: 0h TX_BYP_IN_DEFAULT (default)
12:11	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_MODE: 0h TX_BYP_MODE_FALSE (default) 1h TX_BYP_MODE_TRUE
10:9	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_MODE: 0h RX_BYP_MODE_FALSE (default) 1h RX_BYP_MODE_TRUE
8	R	0	Reserved
7	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_EN: 0h RX_BYP_EN_FALSE (default) 1h RX_BYP_EN_TRUE
6	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_DIR: 0h RX_BYP_DIR_FALSE (default) 1h RX_BYP_DIR_TRUE
5	R	0	Reserved
4	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_RX_BYP_OUT: 0h RX_BYP_OUT_FALSE (default) 1h RX_BYP_OUT_TRUE
3	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_EN: 0h TX_BYP_EN_FALSE (default) 1h TX_BYP_EN_TRUE
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_DIR: 0h TX_BYP_DIR_FALSE (default) 1h TX_BYP_DIR_TRUE
1:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL6_TX_BYP_OUT: 0h TX_BYP_OUT_FALSE (default) 1h TX_BYP_OUT_TRUE

T_SATA<0>_CHX_PHY_CTRL7

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6c0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_WANDER_GEN3: 0h RX_WANDER_GEN3_DEFAULT (default)
19:18	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_TERM_CNTL_GEN3: 0h RX_TERM_CNTL_GEN3_DEFAULT (default)
17:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_TX_TERM_CNTL_GEN3: 0h TX_TERM_CNTL_GEN3_DEFAULT (default)
15:12	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_WANDER_GEN2: 0h RX_WANDER_GEN2_DEFAULT (default)
11:10	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_TERM_CNTL_GEN2: 0h RX_TERM_CNTL_GEN2_DEFAULT (default)

Bit	R/W	Reset	Description
9:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_TX_TERM_CNTL_GEN2: 0h TX_TERM_CNTL_GEN2_DEFAULT (default)
7:4	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_WANDER_GEN1: 0h RX_WANDER_GEN1_DEFAULT (default)
3:2	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_RX_TERM_CNTL_GEN1: 0h RX_TERM_CNTL_GEN1_DEFAULT (default)
1:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL7_TX_TERM_CNTL_GEN1: 0h TX_TERM_CNTL_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL8

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6c4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21:16	R	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_OUT: 0h RX_QEYE_OUT_DEFAULT (default)
15:9	R	0	Reserved
8	R/W	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_EN_GEN3: 0h RX_QEYE_EN_GEN3_DEFAULT (default)
7:5	R	0	Reserved
4	R/W	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_EN_GEN2: 0h RX_QEYE_EN_GEN2_DEFAULT (default)
3:1	R	0	Reserved
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL8_RX_QEYE_EN_GEN1: 0h RX_QEYE_EN_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL9

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6c8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_MISC_TEST: 0h MISC_TEST_DEFAULT (default)
15:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_MISC_OUT_SEL: 0h MISC_OUT_SEL_DEFAULT (default)
7:3	R	0	Reserved
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_DFE_RESET: 0h DFE_RESET_DEFAULT (default)
1	R	0	T_SATA<0>_CHX_PHY_CTRL9_DFE_TRAIN_DONE: 0h DFE_TRAIN_DONE_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL9_DFE_TRAIN_EN: 0h DFE_TRAIN_EN_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL10

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6cc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_CNTL: 0h EOM_CNTL_DEFAULT (default)
15:9	R	0	Reserved
8	R/W	0	T_SATA<0>_CHX_PHY_CTRL10_OOB_RX_CAL_EN: 0h OOB_RX_CAL_EN_DEFAULT (default)
7:3	R	0	Reserved
2	R/W	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_EN: 0h EOM_EN_DEFAULT (default)
1	R	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_TRAIN_DONE: 0h EOM_TRAIN_DONE_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHX_PHY_CTRL10_EOM_TRAIN_EN: 0h EOM_TRAIN_EN_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL11

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6d0 | Read/Write: R/W | Reset: 0x008F000F

Bit	R/W	Reset	Description
31:16	R/W	8Fh	T_SATA<0>_CHX_PHY_CTRL11_GEN2_RX_EQ: 8Fh GEN2_RX_EQ_DEFAULT (default) 4Fh GEN2_RX_EQ_DEFAULT2 0Fh GEN2_RX_EQ_DEFAULT3 CDh GEN2_RX_EQ_DEFAULT4
15:0	R/W	Fh	T_SATA<0>_CHX_PHY_CTRL11_GEN1_RX_EQ: Fh GEN1_RX_EQ_DEFAULT (default) Fh GEN1_RX_EQ_DEFAULT2 Fh GEN1_RX_EQ_DEFAULT3 Fh GEN1_RX_EQ_DEFAULT4

T_SATA<0>_CHX_PHY_CTRL12

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6d4 | Read/Write: R/W | Reset: 0x000000FD

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:0	R/W	FDh	T_SATA<0>_CHX_PHY_CTRL12_GEN3_RX_EQ: FDh GEN3_RX_EQ_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL13

SATA PHY Control Register

This register contains various bits to control the PHY.

Offset: 0x6d8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL13_RX_IQ_CTRL_GEN3: 0h RX_IQ_CTRL_GEN3_DEFAULT (default)
19:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL13_RX_IQ_CTRL_GEN2: 0h RX_IQ_CTRL_GEN2_DEFAULT (default)

Bit	R/W	Reset	Description
15:12	R/W	0	T_SATA<0>_CHX_PHY_CTRL13_RX_IQ_CTRL_GEN1: 0h RX_IQ_CTRL_GEN1_DEFAULT (default)
11:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL13_CDR_TEST: 0h CDR_TEST_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL14

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6dc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL14_DFE_CNTL_GEN1: 0h DFE_CNTL_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL15

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6e0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
23:20	R/W	0	T_SATA<0>_CHX_PHY_CTRL15_RX_FELS_GEN3: 0h RX_FELS_GEN3_DEFAULT (default)
19:16	R/W	0	T_SATA<0>_CHX_PHY_CTRL15_RX_FELS_GEN2: 0h RX_FELS_GEN2_DEFAULT (default)
15:12	R/W	0	T_SATA<0>_CHX_PHY_CTRL15_RX_FELS_GEN1: 0h RX_FELS_GEN1_DEFAULT (default)
11:8	R/W	0	T_SATA<0>_CHX_PHY_CTRL15_DRV_SLEW_GEN3: 0h DRV_SLEW_GEN3_DEFAULT (default)
7:4	R/W	0	T_SATA<0>_CHX_PHY_CTRL15_DRV_SLEW_GEN2: 0h DRV_SLEW_GEN2_DEFAULT (default)
3:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL15_DRV_SLEW_GEN1: 0h DRV_SLEW_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL16

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6e4 | Read/Write: R/W | Reset: 0x00050505

Bit	R/W	Reset	Description
23:16	R/W	5h	T_SATA<0>_CHX_PHY_CTRL16_RX_PI_CTRL_GEN3: 5h RX_PI_CTRL_GEN3_DEFAULT (default)
15:8	R/W	5h	T_SATA<0>_CHX_PHY_CTRL16_RX_PI_CTRL_GEN2: 5h RX_PI_CTRL_GEN2_DEFAULT (default)
7:0	R/W	5h	T_SATA<0>_CHX_PHY_CTRL16_RX_PI_CTRL_GEN1: 5h RX_PI_CTRL_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL17

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6e8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL17_RX_EQ_CTRL_L_GEN1: 0h RX_EQ_CTRL_L_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL18

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6ec | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL18_RX_EQ_CTRL_L_GEN2: 0h RX_EQ_CTRL_L_GEN2_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL19

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6f0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL19_RX_EQ_CTRL_L_GEN3: 0h RX_EQ_CTRL_L_GEN3_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL20

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6f4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL20_RX_EQ_CTRL_H_GEN1: 0h RX_EQ_CTRL_H_GEN1_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL21

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6f8 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL21_RX_EQ_CTRL_H_GEN2: 0h RX_EQ_CTRL_H_GEN2_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL22

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x6fc | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:0	R/W	0	T_SATA<0>_CHX_PHY_CTRL22_RX_EQ_CTRL_H_GEN3: 0h RX_EQ_CTRL_H_GEN3_DEFAULT (default)

T_SATA<0>_CHXCFG3

Serial ATA Control Register

This register is used to send the SATA BIST activate command to the SATA device. The BIST code (byte) is loaded then a '1' is written to bit zero.

Offset: 0x700 | Read/Write: R/W | Reset: 0xXXXX0000

Bit	R/W	Reset	Description
31:16	R	None	T_SATA<0>_CHXCFG3_CHX_PRBS_ERROR_CNT: 0h CHX_PRBS_ERROR_CNT_0
15:8	R/W	0	T_SATA<0>_CHXCFG3_CHX_BIST_CODE: NOTE: this field is reset by Cold Reset 0h CHX_BIST_CODE_DEFAULT (default)
7:2	R	0	Reserved
1	R	0	T_SATA<0>_CHXCFG3_CHX_BIST_STAT: NOTE: this field is reset by Cold Reset 1h CHX_BIST_STAT_BUSY 0h CHX_BIST_STAT_NOT_BUSY (default)
0	RW1C	0	T_SATA<0>_CHXCFG3_CHX_BIST_SEND: NOTE: this field is reset by Cold Reset 1h CHX_BIST_SEND_NOW 0h CHX_BIST_SEND_INIT (default)

T_SATA<0>_CHXCFG4_CHX

Offset: 0x704 | Read/Write: R/W | Reset: 0x0000000a

Bit	R/W	Reset	Description
31:13	R	0	Reserved
12	R/W	0	T_SATA<0>_CHXCFG4_CHX_SW_COMWAKE_PI: NOTE: this field is reset by Cold Reset 0h SW_COMWAKE_PI_INIT (default)
11:8	R/W	0	T_SATA<0>_CHXCFG4_CHX_PHY_ALIGN_NUM_CNT: The PHY_ALIGN_NUM_CNT field configures the number of ALIGN primitive pairs that are inserted by the phy interface block into the output stream. The field value represents a number which is one less than the number of ALIGN pairs to send. So, for e.g., a value of 0 (default) would send 1 ALIGN pair (i.e 2 ALIGNs) every PHY_ALIGN_DWORD_CNT DWORDs. In other words, every PHY_ALIGN_DWORD_CNT DWORDs, the phy-interface sends out (PHY_ALIGN_NUM_CNT+1)*2 ALIGNs NOTE: this field is reset by Cold Reset 0h PHY_ALIGN_NUM_CNT_INIT (default)
7:0	R/W	0Ah	T_SATA<0>_CHXCFG4_CHX_PHY_ALIGN_DWORD_CNT: The PHY_ALIGN_DWORD_CNT field configures the number of DWORDs sent before the required (PHY_ALIGN_NUM_CNT + 1) number of ALIGN primitive pairs are inserted by the phy-interface block into the output stream. For example, the default PHY_ALIGN_DWORD_CNT setting of 254 and default PHY_ALIGN_NUM_CNT of 0 implies an output stream of 254 DWORDs, 2 ALIGNs, 254 DWORDs, 2 ALIGNs, and so on. To disable insertion of ALIGN primitives, set this field to zero. NOTE: this field is reset by Cold Reset Ah PHY_ALIGN_DWORD_CNT_INIT (default)

T_SATA<0>_PRBS_CHX

This register contains control bits for the IOBIST PRBS generator and checker. Setting PI_LOOPBACK will select the PRBS as the TX data source. Setting PRBS_EN will cause the PRBS to start running. PRBS_SEED contains the 10-bit initial value for the PRBS.

Offset: 0x714 | Read/Write: R/W | Reset: 0xXXXXX001

Bit	R/W	Reset	Description
31:16	R	None	T_SATA<0>_PRBS_CHX_ERROR_COUNT:
15	R	None	T_SATA<0>_PRBS_CHX_LOCKED:
14	R	None	T_SATA<0>_PRBS_CHX_ERROR:

Bit	R/W	Reset	Description
13	R	0	Reserved
12	R/W	1	T_SATA<0>_PRBS_CHX_HOT_RESET: NOTE: this field is reset by Cold Reset 1h HOT_RESET_DEFAULT (default)
11	R	0h	T_SATA<0>_PRBS_CHX_RSVD: 0h RSVD_DEFAULT (default)
10	R/W	0	T_SATA<0>_PRBS_CHX_PI_LOOPBACK: NOTE: this field is reset by Cold Reset 0h PI_LOOPBACK_DEFAULT (default)
9:0	R/W	1	T_SATA<0>_PRBS_CHX_SEED: NOTE: this field is reset by Cold Reset 1h SEED_DEFAULT (default)

T_SATA<0>_CHX_PHY_CTRL23

SATA PHY Control Register

This register contains various bits to control the PHY

Offset: 0x718 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
16	R	0	T_SATA<0>_CHX_PHY_CTRL23_RX_EOM_DONE: 0h RX_EOM_DONE_DEFAULT (default)
15:0	R	0	T_SATA<0>_CHX_PHY_CTRL23_RX_EOM_STATUS: 0h RX_EOM_STATUS_DEFAULT (default)

T_SATA<0>_CHX_LINK0

Offset: 0x750 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:3	R	0h	T_SATA<0>_CHX_LINK0_RSVD_0: 0h RSVD_0_VAL
2	R/W	1	T_SATA<0>_CHX_LINK0_LANE_IDDQ_ON_OFFLINE: SW writes this bit for iddq NOTE: this field is reset by Cold Reset 1h LANE_IDDQ_ON_OFFLINE_YES (default) 0h LANE_IDDQ_ON_OFFLINE_NO
1	R/W	1	T_SATA<0>_CHX_LINK0_CONT_DISABLE: CONT_DISABLE is used to disable using CONT primitive in the SATA tx. This bit is provided for each channel. NOTE: this field is reset by Cold Reset 0h CONT_DISABLE_NO 1h CONT_DISABLE_YES 1h CONT_DISABLE_DEFAULT (default)
0	R/W	0	T_SATA<0>_CHX_LINK0_SCRAM_DIS: SCRAM_DIS is used to disable data scrambling in the SATA Link layer. This bit is provided for each channel. NOTE: this field is reset by Cold Reset 0h SCRAM_DIS_NO 1h SCRAM_DIS_YES 0h SCRAM_DIS_DEFAULT (default)

T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR

This register is the backdoor register for PXTFD of the PSM registers section. Used for the purpose of Power Ungating - SAVE and RESTORE vis BKDOOR. Need to write CHX_INDEX to select the channel specific register.

Offset: 0x790 | Read/Write: R/W | Reset: 0xXXXX007F

Bit	Reset	R/W	Description
31:16	0h	R	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_RSVD: 0h RSVD_00
15:8	0	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_ERR: NOTE: this field is reset by Cold Reset 0h ERR_00 (default)
7	0	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_BSY: 0h STS_BSY_CLEAR (default) 1h STS_BSY_SET
6	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_DRDY: 0h STS_DRDY_CLEAR 1h STS_DRDY_SET (default)
5	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_DF: 0h STS_DF_CLEAR 1h STS_DF_SET (default)
4	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_CS: 0h STS_CS_CLEAR 1h STS_CS_SET (default)
3	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_DRQ: 0h STS_DRQ_CLEAR 1h STS_DRQ_SET (default)
2:1	3h	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_2_1: 3h STS_2_1_INIT (default)
0	1	R/W	T_SATA<0>_CHX_AHCI_PORT_PXTFD_BKDR_STS_ERR: 0h STS_ERR_CLEAR 1h STS_ERR_SET (default)

T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR

This register is the backdoor register for PXSIG of the PSM registers. Used for the purpose of Power Ungating - SAVE and RESTORE vis BKDOOR. Need to write CHX_INDEX to select the channel specific register.

Offset: 0x794 | Read/Write: R/W | Reset: 0xFFFFFFFF

Bit	R/W	Reset	Description
31:24	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_LBA_HIGH: FFh LBA_HIGH_INIT (default) 0h LBA_HIGH_00
23:16	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_LBA_MID: FFh LBA_MID_INIT (default) 0h LBA_MID_00
15:8	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_LBA_LOW: FFh LBA_LOW_INIT (default) 0h LBA_LOW_00
7:0	R/W	FFh	T_SATA<0>_CHX_AHCI_PORT_PXSIG_BKDR_SECTOR_CNT: FFh SECTOR_CNT_INIT (default) 0h SECTOR_CNT_00

T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR

This register is the backdoor register for PXSSTS_SPD of the PSM registers. Only STS_DPD is restored by SW, else will be driven by the HW. Used for the purpose of Power Ungating - SAVE and RESTORE vis BKDOOR. Need to write CHX_INDEX to select the channel specific register.

Offset: 0x798 | Read/Write: R/W | Reset: 0xXXXXX000

Bit	R/W	Reset	Description
31:12	R	0h	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_RSVD_31_12: 0h RSVD_31_12_VAL

Bit	R/W	Reset	Description
11:8	R/W	0	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_IPM: 0h IPM_NO_DEV (default) 6h IPM_SLUMBER
7:4	R/W	0	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_SPD: 0h SPD_NO_DEV (default) 1h SPD_GEN1 2h SPD_GEN2 3h SPD_GEN3
3:0	R/W	0	T_SATA<0>_CHX_AHCI_PORT_PXSSTS_BKDR_DET: 0h DET_NO_DEV (default)

T_SATA<0>_CHX_GLUE

Offset: 0x7f0 | Read/Write: R/W | Reset: 0xXXXXXXXX

Bit	R/W	Reset	Description
31:1	R	0h	T_SATA<0>_CHX_GLUE_RSVD_0: 0h RSVD_0_VAL
0	R/W	1	T_SATA<0>_CHX_GLUE_ATAPI_BLINK_EN: qualifier for led_active signal NOTE: this field is reset by Cold Reset 0h ATAPI_BLINK_EN_0 1h ATAPI_BLINK_EN_1 1h ATAPI_BLINK_EN_DEFAULT (default)

T_SATA<0>_INTR

Serial ATA Reserved Register

Offset: 0xac0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31	R	0	T_SATA<0>_INTR_SEC_INTR: SEC_INTR is a status bit for Secondary Interrupt pending. Used for debug and emulation purpose. 1h SEC_INTR_PENDING 0h SEC_INTR_NOT_PENDING (default)
30	R	0	T_SATA<0>_INTR_PRI_INTR: PRI_INTR is a status bit for Primary Interrupt pending. Used for debug and emulation purpose. 1h PRI_INTR_PENDING 0h PRI_INTR_NOT_PENDING (default)
29:0	R	0	Reserved

T_SATA<0>_EMU1

Serial ATA Control Register

Offset: 0xc00 | Read/Write: R/W | Reset: 0x42060F0X

Bit	R/W	Reset	Description
31:28	R/W	4h	T_SATA<0>_EMU1_PHY_UART_TIMEOUT: NOTE: this field is reset by Cold Reset 4h PHY_UART_TIMEOUT_INIT (default)
27	R	0	Reserved
26	R/W	0	T_SATA<0>_EMU1_PHY_USE_RBC1: NOTE: this field is reset by Cold Reset 0h PHY_USE_RBC1_NO (default) 1h PHY_USE_RBC1_YES
25	R/W	1	T_SATA<0>_EMU1_PHY_ABSORB_ALIGN_PRIM: NOTE: this field is reset by Cold Reset 0h PHY_ABSORB_ALIGN_PRIM_NO 1h PHY_ABSORB_ALIGN_PRIM_YES (default)

Bit	R/W	Reset	Description
24	R/W	0	T_SATA<0>_EMU1_PHY_BYPASS_EN: NOTE: this field is reset by Cold Reset 0h PHY_BYPASS_EN_NO (default) 1h PHY_BYPASS_EN_YES
23:16	R/W	06h	T_SATA<0>_EMU1_PHY_UART_SAMPLE: NOTE: this field is reset by Cold Reset 6h PHY_UART_SAMPLE_DEFAULT (default)
15:8	R/W	0Fh	T_SATA<0>_EMU1_PHY_UART_DIV: NOTE: this field is reset by Cold Reset Fh PHY_UART_DIV_DEFAULT (default)
7	R	0	Reserved
6	R/W	0	T_SATA<0>_EMU1_PHY_PM_EN: NOTE: this field is reset by Cold Reset 0h PHY_PM_EN_NO (default) 1h PHY_PM_EN_YES
5	R/W	0	T_SATA<0>_EMU1_PHY_TXCLK_EARLY: 0h PHY_TXCLK_EARLY_NO (default) 1h PHY_TXCLK_EARLY_YES
4	R/W	0	T_SATA<0>_EMU1_PHY_DATA_EARLY: NOTE: this field is reset by Cold Reset 0h PHY_DATA_EARLY_NO (default) 1h PHY_DATA_EARLY_YES
3:1	R	Unknown	T_SATA<0>_EMU1_PHY_SEL: 0h PHY_SEL_NOTHING 1h PHY_SEL_MARVELL 2h PHY_SEL_SI 3h PHY_SEL_NVDA_EXT 4h PHY_SEL_NVDA_INT
0	R/W	1	T_SATA<0>_EMU1_RESET_ON_COMRESET: NOTE: this field is reset by Cold Reset 1h RESET_ON_COMRESET_YES (default) 0h RESET_ON_COMRESET_NO

T_SATA<0>_EMU2

Serial ATA Backdoor Class Code Register

This register changes the SATA device PCI class code register via a backdoor write. This could be used by SBIOS to update the class code before OS sees it.

Offset: 0xc04 | Read/Write: R/W | Reset: 0x000XXXXX

Bit	R/W	Reset	Description
31	R/W	0	T_SATA<0>_EMU2_RETRY_CTL_FIS_SRST: NOTE: this field is reset by Cold Reset 0h RETRY_CTL_FIS_SRST_INIT (default)
30:27	R/W	0	T_SATA<0>_EMU2_AHCI_DEBUG_PORT_SEL_RESERVED: NOTE: this field is reset by Cold Reset 0h AHCI_DEBUG_PORT_SEL_RESERVED_ZERO (default)
26:24	R/W	0	T_SATA<0>_EMU2_AHCI_DEBUG_PORT_SEL: These bits will be used to select the particular port to debug. It is a zero based number used to select the port. NOTE: this field is reset by Cold Reset 0h AHCI_DEBUG_PORT_SEL_ZERO (default) 1h AHCI_DEBUG_PORT_SEL_ONE 2h AHCI_DEBUG_PORT_SEL_TWO 3h AHCI_DEBUG_PORT_SEL_THREE
23:17	R	0	Reserved
16:0	R	0h	T_SATA<0>_EMU2_RSVD: 0h RSVD_INIT

33.9.5.3 DMA Control Registers

I/O Mapped Registers behind a BAR.

T_SATA_PRI_CMD

Bus Master SATA Primary Channel Command Register

The Primary command register allows host to control the SATA primary channel bus mastering.

Offset: 0x000 | Read/Write: R/W | Reset: 0x0X

Bit	R/W	Reset	Description
7:4	R	0	Reserved
3	R/W	0h	T_SATA_PRI_CMD_BM_RW: Read or Write Control. This bit sets the direction of the bus master transfer. When set to zero, PCI bus master reads are performed. When set to one, PCI bus master writes are performed. This bit must NOT be changed when the bus master function is active. 0h BM_RW_READ (default) 1h BM_RW_WRITE
2:1	R	0	Reserved
0	RW1C	None	T_SATA_PRI_CMD_BM_SS: Start/Stop Bus Master. Writing a '1' to this bit enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from a zero to a one. The controller will transfer data between the SATA device and memory only when this bit is set. Master operation can be halted by writing a '0' to this bit. All state information is lost when a '0' is written; Master mode operation cannot be stopped and then resumed. If this bit is reset while bus master operation is still active (i.e., the Bus Master SATA Active bit of the Bus Master SATA Status register for that SATA channel is set) and the drive has not yet finished its data transfer (The Interrupt bit in the Bus Master SATA Status register for that SATA channel is not set), the bus master command is said to be aborted and data transferred from the drive may be discarded before being written to system memory. This bit is intended to be reset after the data transfer is completed, as indicated by either the Bus Master SATA Active bit or the Interrupt bit of the Bus Master SATA Status register for that SATA channel being set, or both. 0h BM_SS_STOP 1h BM_SS_START

T_SATA_PRI_STS

Bus Master SATA Primary Channel Status Register

The Primary channel status registers allows host to check the capabilities and the status of Primary channel bus mastering.

Offset: 0x002 | Read/Write: R/W | Reset: 0xXX

Bit	R/W	Reset	Description
7	R	0h	T_SATA_PRI_STS_SMPLX: Simplex only. This read-only bit is a 0. This indicates that both bus master channels (primary and secondary) can be operated at the same time. 0h SMPLX_NO 1h SMPLX_YES
6	R/W	0h	T_SATA_PRI_STS_DMA1: Drive 1 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 1 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA1_NO (default) 1h DMA1_YES
5	R/W	0h	T_SATA_PRI_STS_DMA0: Drive 0 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 0 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA0_NO (default) 1h DMA0_YES
4:3	R	0	Reserved

Bit	R/W	Reset	Description
2	RW1C	0h	T_SATA_PRI_STS_INTR: Interrupt. This bit is set by the rising edge of the SATA interrupt line. This bit is cleared when a '1' is written to it by software. Software can use this bit to determine if an SATA device has asserted its interrupt line. When this bit is read as a one, all data transferred from the drive is visible in system memory. 0h INTR_NO (default) 1h INTR_YES 1h INTR_CLEAR
1	R	0h	T_SATA_PRI_STS_ERR: Error status '0' = No error '1' = Error occurred 0h ERR_NO 1h ERR_YES
0	R	0h	T_SATA_PRI_STS_BMSATAA: Bus Master SATA Active. This bit is set when the Start bit is written to the Command register. This bit is cleared when the last transfer for a region is performed, where EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Command register. When this bit is read as a zero, all data transferred from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted. 0h BMSATAA_NACTV (default) 1h BMSATAA_ACTV

T_SATA_PRI_PRD_ADD

Primary Channel Descriptor Table Pointer Register

Offset: 0x004 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:2	R/W	0h	T_SATA_PRI_PRD_ADD_DW: Base address of Descriptor table. Corresponds to A[31:2] 0h DW_0 (default)
1:0	R	0	Reserved

T_SATA_SEC_CMD

Bus Master SATA Secondary Channel Command Register

The Secondary command register allows host to control the SATA Secondary channel bus mastering.

Offset: 0x008 | Read/Write: R/W | Reset: 0x0X

Bit	R/W	Reset	Description
7:4	R	0	Reserved
3	R/W	0h	T_SATA_SEC_CMD_BM_RW: Read or Write Control. This bit sets the direction of the bus master transfer. When set to zero, PCI bus master reads are performed. When set to one, PCI bus master writes are performed. This bit must NOT be changed when the bus master function is active. 0h BM_RW_READ (default) 1h BM_RW_WRITE
2:1	R	0	Reserved
0	RW1C	None	T_SATA_SEC_CMD_BM_SS: Start/Stop Bus Master. Writing a '1' to this bit enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from a zero to a one. The controller will transfer data between the SATA device and memory only when this bit is set. Master operation can be halted by writing a '0' to this bit. All state information is lost when a '0' is written; Master mode operation cannot be stopped and then resumed. If this bit is reset while bus master operation is still active (i.e., the Bus Master SATA Active bit of the Bus Master SATA Status register for that SATA channel is set) and the drive has not yet finished its data transfer (The Interrupt bit in the Bus Master SATA Status register for that SATA channel is not set), the bus master command is said to be aborted and data transferred from the drive may be discarded before being written to system memory. This bit is intended to be reset after the data transfer is completed, as indicated by either the Bus Master SATA Active bit or the Interrupt bit of the Bus Master SATA Status register for that SATA channel being set, or both. 0h BM_SS_STOP 1h BM_SS_START

T_SATA_SEC_STS

Bus Master SATA Secondary Channel Status Register

The Secondary channel status registers allows host to check the capabilities and the status of Secondary channel Bus mastering

Offset: 0x00a | Read/Write: R/W | Reset: 0xXX

Bit	R/W	Reset	Description
7	R	0h	T_SATA_SEC_STS_SMPLX: Simplex only. This read-only bit is a 0. This indicates that both bus master channels (primary and secondary) can be operated at the same time. 0h SMPLX_NO 1h SMPLX_YES
6	R/W	0h	T_SATA_SEC_STS_DMA1: Drive 1 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 1 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA1_NO (default) 1h DMA1_YES
5	R/W	0h	T_SATA_SEC_STS_DMA0: Drive 0 DMA Capable. This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 0 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance. 0h DMA0_NO (default) 1h DMA0_YES
4:3	R	0	Reserved
2	RW1C	0h	T_SATA_SEC_STS_INTR: Interrupt. This bit is set by the rising edge of the SATA interrupt line. This bit is cleared when a '1' is written to it by software. Software can use this bit to determine if an SATA device has asserted its interrupt line. When this bit is read as a one, all data transferred from the drive is visible in system memory. 0h INTR_NO (default) 1h INTR_YES 1h INTR_CLEAR
1	R	0h	T_SATA_SEC_STS_ERR: Error status '0' = No error '1' = Error occurred 0h ERR_NO 1h ERR_YES
0	R	0h	T_SATA_SEC_STS_BMSATAA: Bus Master SATA Active. This bit is set when the Start bit is written to the Command register. This bit is cleared when the last transfer for a region is performed, where EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Command register. When this bit is read as a zero, all data transferred from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted. 0h BMSATAA_NACTV (default) 1h BMSATAA_ACTV

T_SATA_SEC_PRD_ADD

Secondary Channel Descriptor Table Pointer Register

Secondary channel PRD table address provides a pointer to base Physical Resource Descriptor Table in the memory. The table is doubleword aligned and must not cross a 64K byte boundary. Also addresses and count in the PRD table are assumed to be aligned are even numbers.

Offset: 0x00c | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:2	R/W	0h	T_SATA_SEC_PRD_ADD_DW: Base address of Descriptor table. Corresponds to A[31:2] 0h DW_0 (default)
1:0	R	0	Reserved

T_SATA_PRI_COMMAND_0

Offset: 0x1f0 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_PRI_COMMAND_0_LBA_LOW: 0h LBA_LOW_INIT (default)
23:16	R/W	0h	T_SATA_PRI_COMMAND_0_SECTOR_COUNT: 0h SECTOR_COUNT_INIT (default)
15:8	R/W	0h	T_SATA_PRI_COMMAND_0_FEATURE_ERR: 0h FEATURE_ERR_INIT (default)
7:0	R/W	0h	T_SATA_PRI_COMMAND_0_DATA: 0h DATA_INIT (default)

T_SATA_PRI_COMMAND_1

Offset: 0x1f4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_PRI_COMMAND_1_COMMAND_STATUS: 0h COMMAND_STATUS_INIT (default)
23:16	R/W	0h	T_SATA_PRI_COMMAND_1_DEVICE: 0h DEVICE_INIT (default)
15:8	R/W	0h	T_SATA_PRI_COMMAND_1_LBA_HIGH: 0h LBA_HIGH_INIT (default)
7:0	R/W	0h	T_SATA_PRI_COMMAND_1_LBA_MID: 0h LBA_MID_INIT (default)

T_SATA_PRI_CONTROL

Offset: 0x3f4 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	T_SATA_PRI_CONTROL_DEVICE_ALTERNATE_STATUS: 0h DEVICE_ALTERNATE_STATUS_INIT (default)
15:0	R	0	Reserved

T_SATA_SEC_COMMAND_0

Offset: 0x170 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_SEC_COMMAND_0_LBA_LOW: 0h LBA_LOW_INIT (default)
23:16	R/W	0h	T_SATA_SEC_COMMAND_0_SECTOR_COUNT: 0h SECTOR_COUNT_INIT (default)
15:8	R/W	0h	T_SATA_SEC_COMMAND_0_FEATURE_ERR: 0h FEATURE_ERR_INIT (default)
7:0	R/W	0h	T_SATA_SEC_COMMAND_0_DATA: 0h DATA_INIT (default)

T_SATA_SEC_COMMAND_1

Offset: 0x174 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R/W	0h	T_SATA_SEC_COMMAND_1_COMMAND_STATUS: 0h COMMAND_STATUS_INIT (default)
23:16	R/W	0h	T_SATA_SEC_COMMAND_1_DEVICE: 0h DEVICE_INIT (default)

Bit	R/W	Reset	Description
15:8	R/W	0h	T_SATA_SEC_COMMAND_1_LBA_HIGH: 0h LBA_HIGH_INIT (default)
7:0	R/W	0h	T_SATA_SEC_COMMAND_1_LBA_MID: 0h LBA_MID_INIT (default)

T_SATA_SEC_CONTROL

Offset: 0x374 | Read/Write: R/W | Reset: 0x00000000

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	T_SATA_SEC_CONTROL_DEVICE_ALTERNATE_STATUS: 0h DEVICE_ALTERNATE_STATUS_INIT (default)
15:0	R	0	Reserved

33.9.6 SATA Aux Registers

33.9.6.1 SATA_AUX_MISC_CNTL_1_0

MISC_CNTL_1 Register

Offset: 0x1108 | Read/Write: R/W | Reset: 0x00016XX0 (0bxxxxxxxxxxxxxx0010110xxx00xx00000)

Bit	R/W	Reset	Description
18	RW	0x0	AUX_RX_IDLE_STATUS_MASK: Set this bit to mask the aux_rx_idle_status input to the SATA core to 0. 0 = Do not mask the rx_stat_idle input to the SATA core 1 = Mask the rx_stat_idle input to the SATA core to 0. 0 = DISABLE 1 = ENABLE
18	RW	0x0	AUX_OR_CORE_IDLE_STATUS_SEL: This bit is used to select the rx_idle status from either AUX or CORE for interrupt generation. 0 = Take the AUX idle status for interrupt generation 1 = Take the CORE idle status for interrupt generation
13	RW	0x1	SDS_SUPPORT: Set this bit to set the AHCI's CAP2.SAS register bit. 0 = Indicates the SATA core does not support device sleep 1 = Indicates the SATA core supports device sleep. 0 = CLEAR 1 = SET
12	RW	0x0	RX_STAT_IDLE_MASK: Set this bit to mask the rx_stat_idle input to the SATA core to 0. 0 = Do not mask the rx_stat_idle input to the SATA core 1 = Mask the rx_stat_idle input to the SATA core to 0. 0 = DISABLE 1 = ENABLE
10:9	RO	X	SATA2IPSM_ST: Indicates whether the SATA link is in partial/slumber modes. Used for debug purposes.
8	RW	0x0	NVA2SATA_OOB_ON_SCONTROL_SPD_WR: If this bit is set to 1, the SATA controller will do an OOB and go through speed negotiation with the drive whenever its SCONTROL_SPD register is written. 0 = NO 1 = YES
7	RW	0x0	NVA2SATA_OOB_ON_POR: Controls whether or not SATA controllers do an OOB sequence automatically when they come out of reset. 0 = NO 1 = YES
6:5	RO	X	L0_RX_IDLE_T_SAX: l0_rx_idle_t value from the SATA controller When the SAX partition is power-gated, this field shows a clamped value. 0 = NORMAL

Bit	R/W	Reset	Description
4:3	RW	0x0	L0_RX_IDLE_T_NPG: Sets the l0_rx_idle_t value for the SATA PHY from apb_misc (non-power-gated logic). Before the SAX partition is power-gated, SATA_l0_rx_idle_t_npg needs to have the same value as SATA_l0_rx_idle_t_sax, then set SATA_l0_rx_idle_t_mux to '1'. This ensures that the SATA PHY is still receiving the correct threshold setting for OOB detection when the SAX is power-gated. After SAX power is restored, the same value is restored to the l0_rx_idle_t register in the SATA controller, then SATA_l0_rx_idle_t_mux can be set back to '0'. 0 = NORMAL
2	RW	0x0	L0_RX_IDLE_T_MUX: Select l0_rx_idle_t driving source for SATA PHY 0: From the SATA controller 1: From apb_misc 0 = FROM_SATA 1 = FROM_APB_MISC
1	RW	0x0	PMU2SATA_ACCLMTR_TRIG: This config bit is used inside SATA to select between the two accelerometer triggers, between the external trigger and the PMU's trigger. 0: External trigger disable 1: External trigger enable 0 = DISABLE 1 = ENABLE
0	RW	0x0	DEVICE_DIS_SATA0: Serial ATA Interface 0 Disable 1 = Internal Serial ATA Interface 0 disabled (Not seen as part of PCI space) 0 = Internal Serial ATA Interface 0 enabled. 0 = ENABLE 1 = DISABLE

33.9.6.2 SATA_AUX_RX_STAT_INT_0

RX_STAT_INT Register

Offset: 0x110c | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxx0xx0xx0xx)

Bit	R/W	Reset	Description
5	RW	0x0	SATA_DEV_ATTEN_INT_DISABLE: SATA device attention interrupt disable 1 = Disable interrupt assertion when dev_atten interrupt status is set 0 = Enable interrupt assertion when dev_atten interrupt status is set 0 = ENABLE 1 = DISABLE
4	RO	X	SATA_DEVICE_ATTENTION: SATA device attention status. 1: device attention input is asserted 0: device attention input is cleared. 0 = NO 1 = YES
3	RO	X	SATA_DEV_ATTEN_INT_STATUS: SATA device attention interrupt status from GPIO. 1 = Device attention interrupt is pending 0 = Device attention interrupt is not pending 0 = NONE 1 = PENDING
2	RW	0x0	SATA_RX_STAT_INT_DISABLE: SATA rx_stat interrupt disable 1 = Disables interrupt assertion when rx_stat interrupt status is set 0 = Enables interrupt assertion when rx_stat interrupt status is set 0 = ENABLE 1 = DISABLE
1	RO	X	SATA_L0_RX_STAT_IDLE: SATA pad L0 rx_stat_idle status 1: Rx path is idle 0: Rx path is active 0 = NO 1 = YES
0	RO	X	SATA_RX_STAT_INT_STATUS: SATA rx_stat interrupt status from the SATA pad 1 = rx_stat interrupt is pending 0 = rx_stat interrupt is not pending 0 = NONE 1 = PENDING

33.9.6.3 SATA_AUX_RX_STAT_SET_0

RX_STAT_SET Register

Offset: 0x1110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
1	0x0	SATA_DEV_ATTEN_INT_SET: SATA dev_atten interrupt is set (this bit is auto-cleared). Set the interrupt status bit to '1' when register is written. The read back value is always '0'.
0	0x0	SATA_RX_STAT_INT_SET: SATA rx_stat interrupt is set (this bit is auto-cleared). Set the interrupt status bit to '1' when register is written. The read back value is always '0'.

33.9.6.4 SATA_AUX_RX_STAT_CLR_0

RX_STAT_CLR Register

Offset: 0x1114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
1	0x0	SATA_DEV_ATTEN_INT_CLR: SATA dev_atten interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.
0	0x0	SATA_RX_STAT_INT_CLR: SATA rx_stat interrupt is cleared (this bit is auto-cleared). Clear the interrupt status bit to '1' when register is written. The read back value is always '0'.

33.9.6.5 SATA_AUX_SPARE_CFG0_0

SPARE_CFG0 Register

Offset: 0x1118 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxx0000000xxxxxxxx)

Bit	R/W	Reset	Description
14	RW	0x0	MDAT_TIMER_AFTER_PG_VALID: Indicates that the MDAT timer value in the above field is to be loaded in the SATA register space.
13:8	RW	0x0	MDAT_TIMER_AFTER_PG: MDAT timer value to be updated by the SW (PEP driver) before power-ungating the SATA controller.
5:0	RO	X	MDAT_TIMER_BEFORE_PG: MDAT timer value loaded from the SATA register space before SATA is power-gated.

33.9.6.6 SATA_AUX_SPARE_CFG1_0

SPARE_CFG1 Register

Offset: 0x111c | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:0	-65536	SPARE: Spare register bits

33.9.6.7 SATA_AUX_PAD_PLL_CTRL_0_0

SATA PAD PLL Control Register

Offset: 0x1120 | Read/Write: R/W | Reset: 0x3X3X0000 (0bxx11xxx0xx11xxx100000x0000000000)

Bit	R/W	Reset	Description
29:28	RW	0x3	PLL1_REFCLK_NDIV
27	RO	X	PLL1_LOCKDET
24	RW	0x0	PLL1_MODE
21:20	RW	0x3	PLL0_REFCLK_NDIV: Select feedback divider for PLL.

Bit	R/W	Reset	Description
19	RO	X	PLL0_LOCKDET: Status signal indicating whether PLL is locked within the desired resolution. Forced high when the PLL is in test modes enabled by either PLL_BYPASS_EN or PLL_EMULATION_ON. 0 = Not locked 1 = Locked 0 = NOT_LOCKED 1 = LOCKED
16	RW	0x1	PLL0_MODE
15:12	RW	0x0	REFCLK_SEL: 00 = Internal CML reference clock 01 = Internal CMOS reference clock 1x = External reference clock 0 = INT_CML 1 = INT_CMOS 2 = EXT
11	RW	0x0	REFCLK_TERM100
9	RW	0x0	PLL_CKBUFPD_OVRD
8	RW	0x0	PLL_CKBUFPD_M
7	RW	0x0	PLL_CKBUFPD_BL
6	RW	0x0	PLL_CKBUFPD_BR
5	RW	0x0	PLL_CKBUFPD_TL
4	RW	0x0	PLL_CKBUFPD_TR
3	RW	0x0	SPARE_BIT_2: Reserved bit.
2	RW	0x0	PLL_EMULATION_RSTN: Digital reset for clock divider during emulation mode, hold low (reset) and release to High to set divider phase. No effect during normal operation. 0 = Reset hold 1 = Reset released/active 0 = ASSERT 1 = DEASSERT
1	RW	0x0	SPARE_BIT_1: Reserved bit.
0	RW	0x0	SPARE_BIT_0: Reserved bit. 0 = LOW 1 = HIGH

33.9.6.8 SATA_AUX_PAD_PLL_CTRL_1_0

Offset: 0x1124 | Read/Write: R/W | Reset: 0xXX441020 (0bxxxxxxxx010001000x01000000100000)

Bit	R/W	Reset	Description
31:24	RO	X	PLL_MISC_OUT: Reserved.
23:20	RW	0x4	PLL1_CP_CNTL: Charge-pump current control for PLL1.
19:16	RW	0x4	PLL0_CP_CNTL: Charge-pump current control for PLL0.
15	RW	0x0	PLL_BYPASS_EN: Bypass PLL serial output clocks with input reference clock.
13	RW	0x0	PLL_EMULATION_ON: Enable clock bypass for emulation mode.
12	RW	0x1	TCLKOUT_EN: Enable test clock output pads, TSTCLKP/N.
11:8	RW	0x0	TCLKOUT_SEL: Select internal clock source to bring out through the TSTCLKP/N pads.
7	RW	0x0	XDIGCLK4P5_EN: Enable XDIGCLK4P5 clock output to core.
6	RW	0x0	REFCLKBUF_EN: Enable REFCLKBUF clock output to core.
5	RW	0x1	TXCLKREF_EN: Enable TXCLKREF clock to core.
4	RW	0x0	TXCLKREF_SEL: Select the post divider for TXCLKREF clock.
3	RW	0x0	XDIGCLK_EN: Enable XDIGCLK output clock.
2:0	RW	0x0	XDIGCLK_SEL: Select the output frequency of XDIGCLK.

33.9.6.9 SATA_AUX_PAD_PLL_CTRL_2_0

Offset: 0x1128 | Read/Write: R/W | Reset: 0x0000XX80 (0b0000xxxx00000000x0xxxxxx1xx00000)

Bit	R/W	Reset	Description
31:28	RW	0x0	PLL_TEMP_CNTL
23:18	RW	0x0	PLL_BW_CNTL
17:16	RW	0x0	PLL_BGAP_CNTL
15	RO	X	RCAL_DONE: Status signal to indicate calibration status.
14	RW	0x0	RCAL_RESET: Reset the resistor calibration logic.
12:8	RO	X	RCAL_VAL: Setting of current active resistor calibration code
7	RW	0x1	RCAL_BYPASS: Bypass resistor calibration logic.
4:0	RW	0x0	RCAL_CODE: Sets resistor calibration code when logic is bypassed.

33.9.6.10 SATA_AUX_PAD_PLL_CTRL_3_0

Offset: 0x112c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PLL_MISC_CNTL

33.9.6.11 SATA_AUX_PAD_L0_AUX_CTRL_0_0

Offset: 0x1130 | Read/Write: R/W | Reset: 0x00000X00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	R/W	Reset	Description
9	RO	X	AUX_RX_IDLE_STATUS
8	RO	X	AUX_TX_RDET_STATUS
5	RW	0x0	AUX_HOLD_EN
4	RW	0x0	AUX_RX_IDLE_MODE
3	RW	0x0	AUX_RX_IDLE_EN
2	RW	0x0	AUX_RX_TERM_EN
1	RW	0x0	AUX_TX_RDET_EN
0	RW	0x0	AUX_TX_TERM_EN

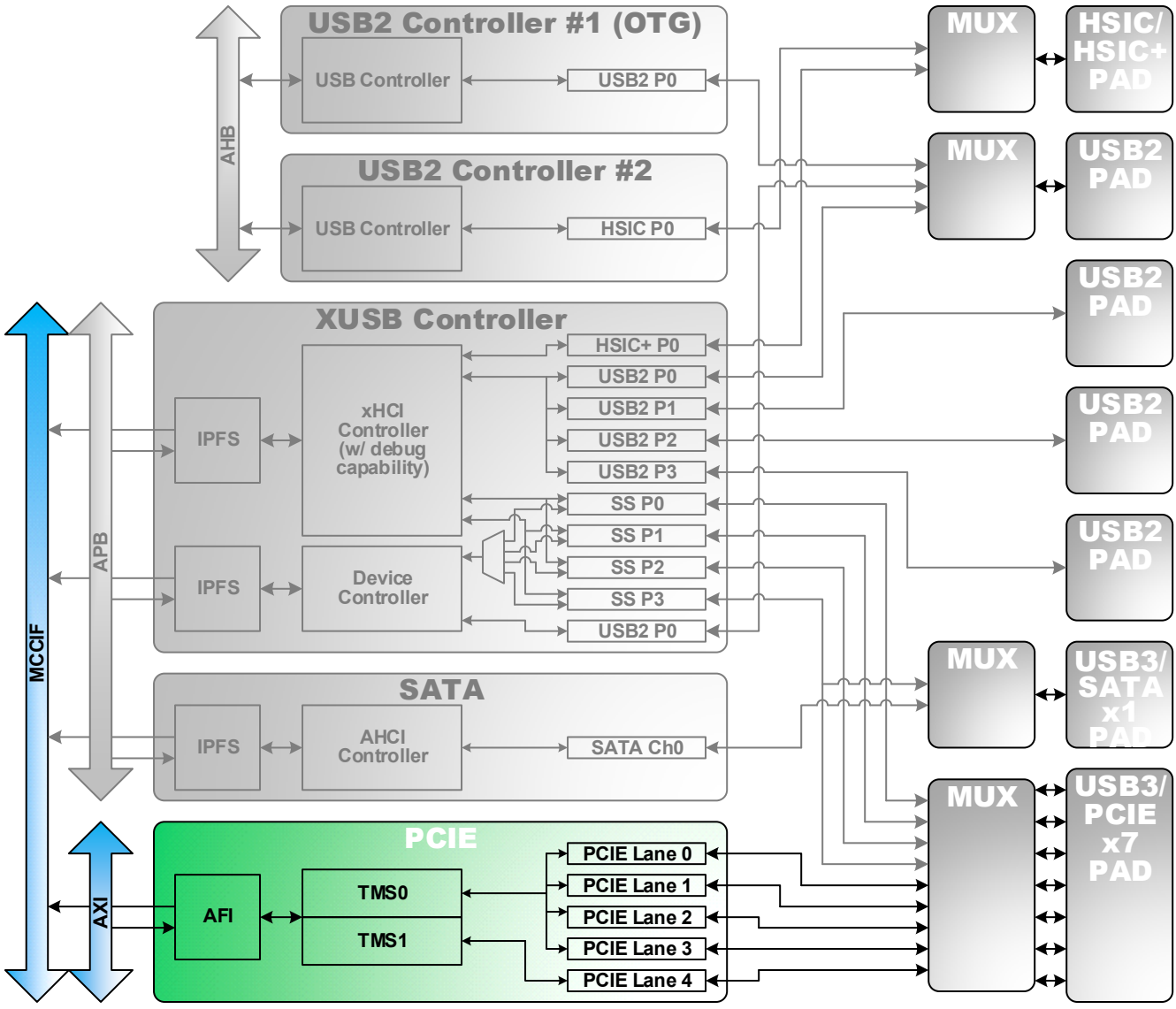
CHAPTER 34: PCI EXPRESS (PCIe) CONTROLLER

Tegra® X1 devices support up to 2 PCIe® root port controllers. The PCIe controllers support the following configurations:

- the first controller 4 lanes; the second 1 lane (x4 + x1)
- the first controller 2 lanes; the second 1 lane (x2 + x1)

Both controllers support upstream and downstream AXI interfaces that serve as the control path from the Tegra X1 device to the external PCIe device. Both PCIe controllers support Gen1 (2.5 GT/s/lane) and Gen2 (5.0 GT/s/lane) speeds.

Figure 157: PCIe Interface



34.1 Supported Configurations

The lane mappings for the configurations are internal to the PCIe root port controller. For supported configurations details, refer to Section 22.7: USB PADCTL in the USB Complex chapter.

34.2 L1 PM Substate Support

The Tegra X1 PCIe controller supports the L1 PM substate for power savings. Tegra X1 processor support all three substates: L1.0, L1.1, and L1.2 as defined in the ECN entitled *L1 PM Substates with CLKREQ* (see www.pcisig.com) L1.0 is equivalent to L1. L1.1 enables turning off EIDLE exit detection circuitry in the PHY, and L1.2 enables turning off both EIDLE exit detection circuitry and common mode voltage regulators.

Tegra X1 processors save more power with L1 PM substate support by turning off EIDLE exit detection circuitry in the PHY (in both L1.1 and L1.2 substates) and common mode voltage regulators in the L1.2 substate.

Latency Tolerance Reporting (LTR) has been added to enhance Power Management substate support.

34.3 Programming Guidelines

34.3.1 Root Port Initialization Sequence

This section describes the PCIe root port initialization. This can be used as reference for the initialization sequence for software and Tegra shell for bring-up.

The initialization sequence involves bringing PCIE partition out of power gating as partitions stay in power gated state after the system cold boots. PLL, clocks, and pads used by PCIE are initialized while AFI and PCIE are under reset, followed by setting up the AFI and PCIE controller. Lastly, the reset to the connected PCIE device is deasserted to allow the link to be initialized where SW can access the connected device for enumeration.

Note: *The format for register writes used in this section is register name followed by field name in square brackets followed by the value: <register name>[field name] value.*

The PCIe root port initialization sequence is summarized in the following steps:

1. Set PCIEXCLK reset
   - CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_PCIEXCLK_RST] 1
2. Set PCIE reset
   - CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_PCIE_RST] 1
3. Set AFI reset
   - CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_AFI_RST] 1
4. Set PEX_USB_UPHY reset
   - CLK_RST_CONTROLLER_RST_DEV_Y_SET_0[SET_PEX_USB_UPHY_RST] 1
5. Set XUSB_PADCTL reset
   - CLK_RST_CONTROLLER_RST_DEV_W_SET_0[SET_XUSB_PADCTL_RST] 1
6. Enable PLLe (refer to Section 22.8.4: Cold Boot, with no Recovery Mode or Boot from USB in the USB Complex chapter)
7. Restore AFI and PCIE power by setting following registers bits
   - APBDEV_PMC_PWRGATE_TOGGLE_0[START] enable
   - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] PCX

     Then confirm the power restore by reading the following fields:
   - APBDEV_PMC_PWRGATE_STATUS_0[PCX] equals ‘ON’
8. AFI and PCIE power clamping is removed by setting the following register fields:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [PCX]

     and then confirm the power clamping removal by reading the flowing field

- APBDEV_PMC_REMOVE_CLAMPING_CMD_0[PCX] equals ‘0’

9. Enable AFI Clock
   - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_AFI] 1
10. Enable PCIe Clock
   - CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_PCIE] 1
11. Clear XUSB_PADCTL reset
   - CLK_RST_CONTROLLER_RST_DEV_W_CLR_0[CLR_XUSB_PADCTL_RST] 1
12. Enable UPHY PLL (refer to Section 22.8.4: Cold Boot, with no Recovery Mode or Boot from USB)
    UPHY reset is cleared in the sequence
13. Disable the IDDQ for all lanes for PCIe tests.
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0] 1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1] 1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2] 1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3] 1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4] 1
14. Clear AUX_MUX_LP0 related bits in register XUSB_PADCTL_ELPG_PROGRAM_1_0
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[AUX_MUX_LP0_CLAMP_EN] 0
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[AUX_MUX_LP0_CLAMP_EN_EARLY] 0
   - XUSB_PADCTL_ELPG_PROGRAM_1_0[AUX_MUX_LP0_VCORE_DOWN] 0
15. Wait 200 us
16. Set Lane IDDQ and SLEEP Override
   - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[TX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[RX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[TX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[RX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[TX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[RX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[TX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[RX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[TX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[RX_PWR_OVRD] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[TX_IDDQ] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[RX_IDDQ] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[TX_IDDQ] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[RX_IDDQ] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[TX_IDDQ] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[RX_IDDQ] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[TX_IDDQ] 1
   - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[RX_IDDQ] 1

- XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[TX_IDDQ] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[RX_IDDQ] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[TX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[RX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[TX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[RX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[TX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[RX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[TX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[RX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[TX_IDDQ_OVRD] 1
- XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[RX_IDDQ_OVRD] 1

17. Update Lane Mux Ownership (illustrative example for x4_x1)
    - XUSB_PADCTL_USB3_PAD_MUX_0 PCIE_PAD_LANE0 0x0
    - XUSB_PADCTL_USB3_PAD_MUX_0 PCIE_PAD_LANE1 0x3
    - XUSB_PADCTL_USB3_PAD_MUX_0 PCIE_PAD_LANE2 0x3
    - XUSB_PADCTL_USB3_PAD_MUX_0 PCIE_PAD_LANE3 0x3
    - XUSB_PADCTL_USB3_PAD_MUX_0 PCIE_PAD_LANE4 0x3
18. Clear Lane IDDQ and SLEEP Override
    - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[TX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[RX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[TX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[RX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[TX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[RX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[TX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[RX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[TX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[RX_PWR_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[TX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P0_CTL_2_0[RX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[TX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P1_CTL_2_0[RX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[TX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P2_CTL_2_0[RX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[TX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P3_CTL_2_0[RX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[TX_IDDQ_OVRD] 0
    - XUSB_PADCTL_UPHY_MISC_PAD_P4_CTL_2_0[RX_IDDQ_OVRD] 0

19. Configure the pin mux
    a. (L1SS disabled)
       PINMUX_AUX_PEX_L0_CLKREQ_N_0[TRISTATE] 1
       PINMUX_AUX_PEX_L1_CLKREQ_N_0[TRISTATE] 1
    b. Configure reset pad
       PINMUX_AUX_PEX_L0_RST_N_0[TRISTATE] 0
       PINMUX_AUX_PEX_L1_RST_N_0[TRISTATE] 0
20. Clear AFI reset
    - CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_AFI_RST] 1
21. Program AFI registers
    a. Enable Gen2 speed
       AFI_FUSE_0[FUSE_PCIE_T0_GEN2_DIS] 0
    b. Enable PCIE devices
       AFI_PCIE_CONFIG_0[PCIEC0_DISABLE_DEVICE] 0
       AFI_PCIE_CONFIG_0[PCIEC1_DISABLE_DEVICE] 0
    c. Program XBAR config (not required for Tegra X1 as this is default value and only supported config)
       AFI_PCIE_CONFIG_0[SM2TMS0_XBAR_CONFIG] 1
    d. Enable FPCI interface
       AFI_CONFIGURATION_0[EN_FPCI] 1
    e. Set register for PLL power down
       AFI_PLLE_CONTROL_0[BYPASS_PADS2PLLE_CONTROL] 0
       AFI_PLLE_CONTROL_0[BYPASS_PCIE2PLLE_CONTROL] 0
       AFI_PLLE_CONTROL_0[PADS2PLLE_CONTROL_EN] 1
       AFI_PLLE_CONTROL_0[PCIE2PLLE_CONTROL_EN] 1
    f. Power up bias pad by register programming
       AFI_PEXBIAS_CTRL_0[PEX_BIAS_PWRD] 0
    g. Override refclk initially as some EP may not assert clkreq on boot and for platform not supporting clock power management this is necessary
       AFI_PEX0_CTRL_0[PEX0_REFCLK_OVERRIDE_EN] 1
       AFI_PEX1_CTRL_0[PEX1_REFCLK_OVERRIDE_EN] 1
    h. Enable refclk and clkreq pad
       AFI_PEX0_CTRL_0[PEX0_REFCLK_EN] 1
       AFI_PEX0_CTRL_0[PEX0_CLKREQ_EN] 0
       AFI_PEX1_CTRL_0[PEX1_REFCLK_EN] 1
       AFI_PEX1_CTRL_0[PEX1_CLKREQ_EN] 0
    i. Assert EP reset
       AFI_PEX0_CTRL_0[PEX0_RST_L] 0
       AFI_PEX1_CTRL_0[PEX1_RST_L] 0
    j. Wait for reset to get applied

k. Release EP reset

AFI_PEX0_CTRL_0[PEX0_RST_L] 1

AFI_PEX1_CTRL_0[PEX1_RST_L] 1

22. Clear PCIe controller reset

- CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_PCIE_RST] 1

23. Program PCIe register

a. Configure refclk pad

PCIE2_PADS_REFCLK_CFG0_0 0x90b890b8

b. Configure UPHY electrical settings

PCIE2_RP_ECTL_2_R1_0[RX_CTLE_1C] 0x000F

PCIE2_RP_ECTL_2_R2_0[RX_CTLE_1C] 0x008F

PCIE2_RP_ECTL_5_R1_0[RX_EQ_CTRL_L_1C] 0x55010000

PCIE2_RP_ECTL_6_R1_0[RX_EQ_CTRL_H_1C] 0x00000001

PCIE2_RP_ECTL_5_R2_0[RX_EQ_CTRL_L_1C] 0x55010000

PCIE2_RP_ECTL_6_R2_0[RX_EQ_CTRL_H_1C] 0x00000001

PCIE2_RP_ECTL_4_R1_0[RX_CDR_CTRL_1C] 0x0067

PCIE2_RP_ECTL_4_R2_0[RX_CDR_CTRL_1C] 0x00C7

PCIE2_RP_ECTL_2_R1_1[RX_CTLE_1C] 0x000F

PCIE2_RP_ECTL_2_R2_1[RX_CTLE_1C] 0x008F

PCIE2_RP_ECTL_5_R1_1[RX_EQ_CTRL_L_1C] 0x55010000

PCIE2_RP_ECTL_6_R1_1[RX_EQ_CTRL_H_1C] 0x00000001

PCIE2_RP_ECTL_5_R2_1[RX_EQ_CTRL_L_1C] 0x55010000

PCIE2_RP_ECTL_6_R2_1[RX_EQ_CTRL_H_1C] 0x00000001

PCIE2_RP_ECTL_4_R1_1[RX_CDR_CTRL_1C] 0x0067

PCIE2_RP_ECTL_4_R2_1[RX_CDR_CTRL_1C] 0x00C7

c. Enable PCA

PCIE2_RP_VEND_CYA2_0 PCA_ENABLE 1

PCIE2_RP_VEND_CYA2_1 PCA_ENABLE 1

d. Enable clock clamping and change threshold for TMS clock clamping

PCIE2_RP_PRIV_MISC_0[TMS_CLK_CLAMP_ENABLE] 1

PCIE2_RP_PRIV_MISC_0[CTLR_CLK_CLAMP_ENABLE] 1

PCIE2_RP_PRIV_MISC_0[TMS_CLK_CLAMP_THRESHOLD] 0xF

PCIE2_RP_PRIV_MISC_0[CTLR_CLK_CLAMP_THRESHOLD] 0xF

PCIE2_RP_PRIV_MISC_1[TMS_CLK_CLAMP_ENABLE] 1

PCIE2_RP_PRIV_MISC_1[CTLR_CLK_CLAMP_ENABLE] 1

PCIE2_RP_PRIV_MISC_0[TMS_CLK_CLAMP_THRESHOLD] 0xF

PCIE2_RP_PRIV_MISC_0[CTLR_CLK_CLAMP_THRESHOLD] 0xF

e. Configure PCIE2_RP_VEND register to enable ASPM

PCIE2_RP_VEND_XP1_0[CYA] 0x4

PCIE2_RP_VEND_XP1_1[CYA] 0x4

f. Set BIST register

PCIE2_RP_VEND_XP_BIST_0 0x10000001

PCIE2_RP_VEND_XP_BIST_1 0x10000001

g. Enable opportunistic update FC and ACK to send out update FC and ACK when link is idle

PCIE2_RP_VEND_XP_0[OPPORTUNISTIC_ACK] 1

PCIE2_RP_VEND_XP_0[OPPORTUNISTIC_UPDATEFC] 1

PCIE2_RP_VEND_XP_1[OPPORTUNISTIC_ACK] 1

PCIE2_RP_VEND_XP_1[OPPORTUNISTIC_UPDATEFC] 1

h. Update flow control threshold for X1 controller

PCIE2_RP_VEND_XP_0[UPDATE_FC_THRESHOLD] 0x60

PCIE2_RP_VEND_XP_1[UPDATE_FC_THRESHOLD] 0x60

i. Set CLKREQ assertion delay

PCIE2_RP_L1_PM_SUBSTATES_1_CYA_0[CHK_CLKREQ_ASSERTED_DLY] 0x4F

PCIE2_RP_L1_PM_SUBSTATES_1_CYA_1[CHK_CLKREQ_ASSERTED_DLY] 0x4F

j. Software tuning for the de-skew retry time

PCIE2_RP_VEND_CYA0_0[DSK_RESET_PULSE_WIDTH] 0x9

PCIE2_RP_VEND_CYA0_1[DSK_RESET_PULSE_WIDTH] 0x9

k. Power saving for sleep/idle

PCIE2_RP_VEND_XP_PAD_PWRDN_0[DISABLED] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_0[DYNAMIC] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_0[L1] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_0[L1_CLKREQ] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_1[DISABLED] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_1[DYNAMIC] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_1[L1] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_1[L1_CLKREQ] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_0[SLEEP_MODE_DYNAMIC] 0x3

PCIE2_RP_VEND_XP_PAD_PWRDN_0[SLEEP_MODE_L1] 0x3

PCIE2_RP_VEND_XP_PAD_PWRDN_0[SLEEP_MODE_L1_CLKREQ] 0x3

PCIE2_RP_VEND_XP_PAD_PWRDN_1[SLEEP_MODE_DYNAMIC] 0x3

PCIE2_RP_VEND_XP_PAD_PWRDN_1[SLEEP_MODE_L1] 0x3

PCIE2_RP_VEND_XP_PAD_PWRDN_1[SLEEP_MODE_L1_CLKREQ] 0x3

PCIE2_RP_VEND_XP_PAD_PWRDN_0[SLEEP_MODE_DYNAMIC] 0

PCIE2_RP_VEND_XP_PAD_PWRDN_0[IDLE_MODE_L1] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_0[IDLE_MODE_L1_CLKREQ] 0

PCIE2_RP_VEND_XP_PAD_PWRDN_1[SLEEP_MODE_DYNAMIC] 0

PCIE2_RP_VEND_XP_PAD_PWRDN_1[IDLE_MODE_L1] 1

PCIE2_RP_VEND_XP_PAD_PWRDN_1[IDLE_MODE_L1_CLKREQ] 0

l. Settings for the 19.2MHz CLK25M clock frequency

PCIE2_RP_TIMEOUT0_0[PAD_PWRUP] 0xa

PCIE2_RP_TIMEOUT0_0[PAD_PWRUP_CM] 0x180

PCIE2_RP_TIMEOUT0_0[PAD_SPDCHNG_GEN2] 0xa

PCIE2_RP_TIMEOUT0_1[PAD_PWRUP] 0xa

PCIE2_RP_TIMEOUT0_1[PAD_PWRUP_CM] 0x180

PCIE2_RP_TIMEOUT0_1[PAD_SPDCHNG_GEN2] 0xa

PCIE2_RP_TIMEOUT1_0[RCVRY_SPD_SUCCESS_EIDLE] 0x10

PCIE2_RP_TIMEOUT1_0[RCVRY_SPD_UNSUCCESS_EIDLE] 0x74

PCIE2_RP_TIMEOUT1_1[RCVRY_SPD_SUCCESS_EIDLE] 0x10

PCIE2_RP_TIMEOUT1_1[RCVRY_SPD_UNSUCCESS_EIDLE] 0x74

PCIE2_RP_XP_REF_0[CPL_TO_OVERRIDE] 0x1

PCIE2_RP_XP_REF_0[CPL_TO_CUSTOM_VALUE] 0x1770

PCIE2_RP_XP_REF_1[CPL_TO_OVERRIDE] 0x1

PCIE2_RP_XP_REF_1[CPL_TO_CUSTOM_VALUE] 0x1770

PCIE2_RP_XP_REF_0[MICROSECOND_LIMIT] 0x14

PCIE2_RP_XP_REF_0[MICROSECOND_ENABLE] 0x1

PCIE2_RP_XP_REF_1[MICROSECOND_LIMIT] 0x14

PCIE2_RP_XP_REF_1[MICROSECOND_ENABLE] 0x1

PCIE2_RP_L1_PM_SUBSTATES_2_CYA_0[T_L1_2_DLY] 0x4D

PCIE2_RP_L1_PM_SUBSTATES_2_CYA_0[MICROSECOND] 0x13

PCIE2_RP_L1_PM_SUBSTATES_2_CYA_0[MICROSECOND_COMP] 0x2

PCIE2_RP_L1_PM_SUBSTATES_2_CYA_1[T_L1_2_DLY] 0x4D

PCIE2_RP_L1_PM_SUBSTATES_2_CYA_1[MICROSECOND] 0x13

PCIE2_RP_L1_PM_SUBSTATES_2_CYA_1[MICROSECOND_COMP] 0x2

PCIE2_RP_L1_PM_SUBSTATES_1_CYA_0[T_PWR_OFF_DLY] 0x26

PCIE2_RP_L1_PM_SUBSTATES_1_CYA_0[CHK_CLKREQ_ASSERTED_DLY] 0x27

PCIE2_RP_L1_PM_SUBSTATES_1_CYA_1[T_PWR_OFF_DLY] 0x26

PCIE2_RP_L1_PM_SUBSTATES_1_CYA_1[CHK_CLKREQ_ASSERTED_DLY] 0x27

24. Read GPIO for the EP card presence and program present map register for the present controllers

- PCIE2_RP_PRIV_MISC_0[PRSNT_MAP] 0xE

- PCIE2_RP_PRIV_MISC_1[PRSNT_MAP] 0xE

25. Clear reset to PCIe device
    - CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_PCIEXCLK_RST] 1
26. Disable PCA
    - PCIE2_RP_VEND_CYA2_0 PCA_ENABLE 0
    - PCIE2_RP_VEND_CYA2_1 PCA_ENABLE 0
27. Poll for DL UP
    - Poll PCIE2_RP_VEND_XP[DL_UP] 1

34.3.2 Enable Advance Error Reporting

PCIe design support Advance Error Reporting. Standard settings used for this feature are as follows:

1. Enable error reporting for all error types
    - PCIE2_RP_DEVICE_CONTROL_STATUS_0[CORR_ERROR_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_0[NON_FATAL_ERROR_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_0[FATAL_ERROR_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_0[UNSUPP_REQ_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_0[ENABLE_RELAXED_ORDERING] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_1[CORR_ERROR_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_1[NON_FATAL_ERROR_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_1[FATAL_ERROR_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_1[UNSUPP_REQ_REPORTING_ENABLE] 1
    - PCIE2_RP_DEVICE_CONTROL_STATUS_1[ENABLE_RELAXED_ORDERING] 1
2. Enable error indication to AFI
    - PCIE2_RP_RCR_0[SERR_COR] 1
    - PCIE2_RP_RCR_0[SERR_NONFAT] 1
    - PCIE2_RP_R1CR_0[SERR_FAT] 1
    - PCIE2_RP_RCR_1[SERR_COR] 1
    - PCIE2_RP_RCR_1[SERR_NONFAT] 1
    - PCIE2_RP_R1CR_1[SERR_FAT] 1
    - PCIE2_RP_INTR_BCR_0[SERR_FORWARD] 1
    - PCIE2_RP_INTR_BCR_1[SERR_FORWARD] 1

34.3.3 Address Space Mapping in AFI and PCIe

Tegra X1 address space is defined based on AXI and there is one single address domain. As well, PCIe inherits three address domains from Legacy PCI and these domains are memory, I/O, and, configuration. For I/O and configuration space, software needs to use SO (Strongly Ordered) or DEV (Device) mem type to guarantee 4 byte transactions.

The BAR mappings in AFI allow conversions from Tegra X1 AXI address space to the HyperTransport address space recognized by PCIe Root Port controllers. PCIe Root Port controllers decode the HyperTransport addresses to generate PCIe requests to different address spaces. AFI provides 8 sets of address mapping registers that allow software to set up address mappings required by the address space conversions. This allows more flexibility for software to allocate and map system addresses to different address spaces under one or more PCIe controllers.

This register defines the starting address in the AXI address space to be mapped to FPCI address.

- AFI_AXI_BAR*_START_0[AXI_BAR*_START]

This register defines the size of the AXI address space to be mapped to FPCI address space.

- AFI_AXI_BAR*_SZ_0[AXI_BAR0_SIZE]

This register defines the starting address space in the FPCI address where AXI address to be mapped to.

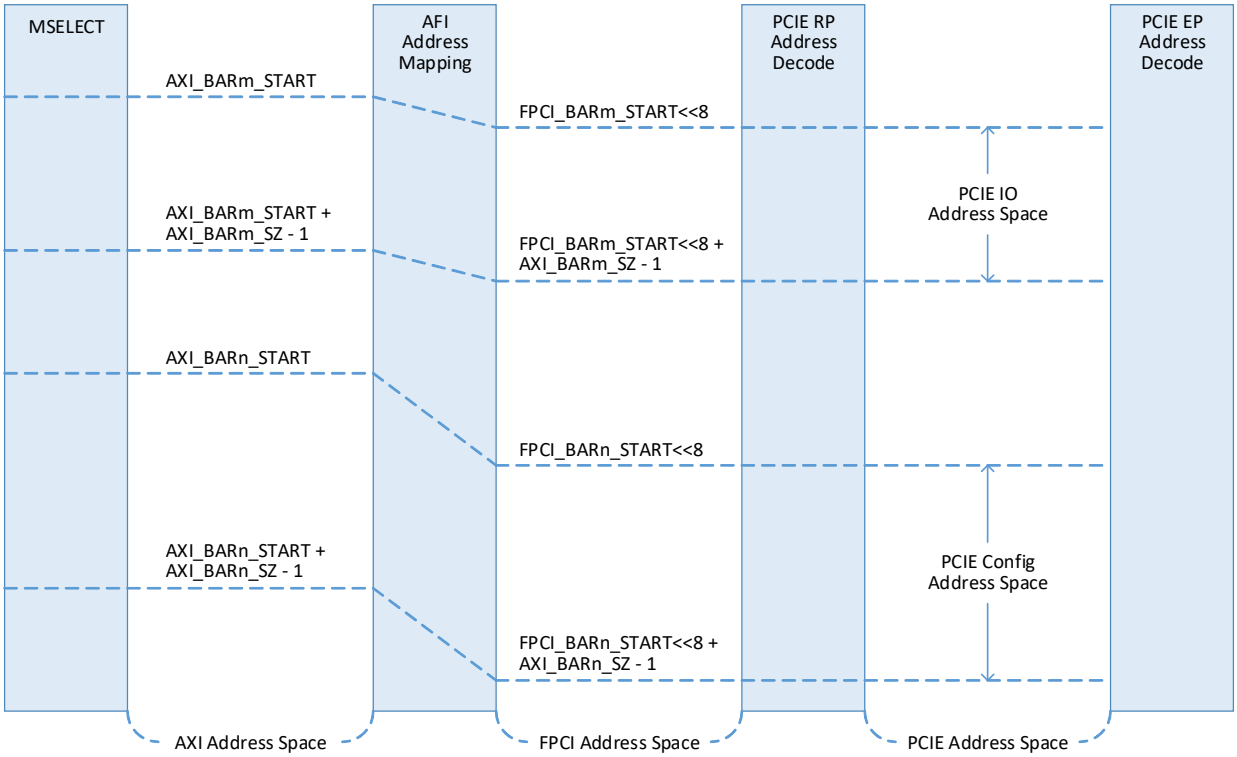
- AFI_FPCI_BAR0_0[FPCI_BAR*_START]

This address defines whether the write requests that are converted to FPCI addresses should also be converted to posted requests.

- AFI_FPCI_BAR0_0[FPCI_BAR*_ACCESS_TYPE]

The following figure illustrates an example of the address space conversion and decoding.

Figure 158: Address Space Conversion and Decoding



The following subsections describe how to set up the address mapping in AFI.

34.3.3.1 PCIE Configuration Requests

The configuration registers of the PCIE Root Port controllers are allocated outside the PCIE configuration address space as listed in Section 34.4: PCIe Registers. The BAR address mapping in AFI is used for defining the Type 1 configuration address space for accessing configuration registers of connected external PCIE endpoints.

For Type 1 configuration address space, AFI_FPCI_BAR0_0[FPCI_BAR*_START] should be set to 0xFE10_000.

As configuration write requests are defined as non-posted, writes in PCIE, AFI_FPCI_BAR0_0[FPCI_BAR*_ACCESS_TYPE] should be set to IO/Config Access.

Closed System

In a closed system where connected external devices are determined by platform configurations of the vendors, the size of the configuration address space can be limited. The equation to calculate the size of the Type 1 config space is:

- 1MB * Number of PCI Bus Allocated

Open System

In an open system where connected external devices can be determined by end users, the size of the configuration address space should be defined as 256 MB. Software performs PCI compliant device discovery and enumeration to identify and set up connected PCIE endpoints.

Since the system interface of the PCIE Root Port controllers follow the HyperTransport definition, SW should generate AXI requests with addresses following the HyperTransport definition of Type 1 configuration requests, where:

- Bus Number[7:0]= AXI_ADDRESS[23:16]
- Device Number[4:0]= AXI_ADDRESS[15:11]
- Function Number[3:0]= AXI_ADDRESS[10:8]
- Register Number[11:0]= {AXI_ADDRESS[27:24], AXI_ADDRESS[7:0]}

34.3.3.2 PCIE I/O Requests

The BAR address mapping in AFI is used for defining the I/O address space for accessing I/O registers of connected external PCIE endpoints.

For I/O address space, AFI_FPCI_BAR0_0[FPCI_BAR*_START] should be set to 0xFDFC_000.

As configuration write requests are defined as non-posted, writes in PCIE, AFI_FPCI_BAR0_0[FPCI_BAR*_ACCESS_TYPE] should be set to IO/Config Access.

Closed System

In a closed system where connected external devices are determined by platform configurations of the vendors, the size of the I/O address space can be limited.

If none of the connected devices require I/O address spaces as identified by their PCI BAR registers, the size of the I/O address space should be set to 0 to disable the I/O address space mapping.

If some of the connected devices require I/O address spaces as identified by their PCI BAR registers, the size of the I/O address space should be set to cover the sum of all I/O spaces required by the devices.

Open System

In an open system where connected external devices can be determined by end users, the size of the I/O address space should be defined as 32 MB. Software performs PCI compliant device discovery and enumeration to identify and set up connected PCIE endpoints.

34.3.3.3 PCIE Memory Requests

While the AXI addresses can be directly used to access the memory mapped registers of the PCIE endpoints, AFI has to perform the conversion of non-posted AXI write requests to the posted PCIE memory write requests. Thus, memory space is also required to be defined in AFI.

For memory address space, AFI_FPCI_BAR0_0[FPCI_BAR*_START] could be set to match AFI_AXI_BAR*_START_0[AXI_BAR*_START].

As memory write requests are defined as posted, writes in PCIE, AFI_FPCI_BAR0_0[FPCI_BAR*_ACCESS_TYPE] should be set to Memory Access.

Closed System

In a closed system where connected external devices are determined by platform configurations of the vendors, the size of the memory address space can be limited. The size of the memory address space should be set to cover the sum of all memory spaces required by the devices as identified by their PCI BAR registers.

Open System

In an open system where connected external devices can be determined by end users, the size of the memory address space should be set to cover the remaining AXI address space allocated to PCIE.

34.3.4 PCIE Hierarchy Enumeration

Following PCI/PCIE specification, the PCIE hierarchy enumeration is performed via depth first search to discover all devices attach to the PCIE controllers and initialize their CFG registers.

The following register bits should be set to enable PCIE to forward DMA requests and accept MMIO and IOIO requests from CPU:

- PCIE2_RP_DEV_CTRL_0[BUS_MASTER] 1’b1
- PCIE2_RP_DEV_CTRL_0[MEMORY_SPACE] 1’b1
- PCIE2_RP_DEV_CTRL_0[IO_SPACE] 1’b1
- PCIE2_RP_DEV_CTRL_1[BUS_MASTER] 1’b1
- PCIE2_RP_DEV_CTRL_1[MEMORY_SPACE] 1’b1
- PCIE2_RP_DEV_CTRL_1[IO_SPACE] 1’b1

The configuration register space of the PCIE hierarchy below each PCIE root port is accesses by using the AFI address mapping to convert the Tegra MMIO addresses to PCIE config addresses. The following PCIE registers should be programmed accordingly for access configuration register spaces below the root ports:

- PCIE2_RP_BN_LT_0[SEC_BUS_NUMBER]
- PCIE2_RP_BN_LT_0[SUB_BUS_NUMBER]
- PCIE2_RP_BN_LT_1[SEC_BUS_NUMBER]
- PCIE2_RP_BN_LT_1[SUB_BUS_NUMBER]

The IOIO register space of the PCIE hierarchy below each PCIE root port is accesses by using the AFI address mapping to convert the Tegra MMIO addresses to PCIE IOIO addresses. The following PCIE registers should be programmed accordingly for access IOIO register spaces below the root ports:

- PCIE2_RP_IO_BL_SS_0[IO_BASE]
- PCIE2_RP_IO_BL_SS_0[IO_LIMIT]
- PCIE2_RP_IO_BL_SS_1[IO_BASE]
- PCIE2_RP_IO_BL_SS_1[IO_LIMIT]

The IOIO register space of the PCIE hierarchy below each PCIE root port is accesses by using the AFI address mapping to shift the Tegra MMIO addresses to PCIE MMIO addresses. The following PCIE registers should be programmed accordingly for access MMIO register spaces below the root ports:

- PCIE2_RP_MEM_BL_0[MEM_BASE]
- PCIE2_RP_MEM_BL_0[MEM_LIMIT]
- PCIE2_RP_PRE_BL_0[PREFETCH_MEM_BASE]
- PCIE2_RP_PRE_BL_0[PREFETCH_MEM_LIMIT]
- PCIE2_RP_MEM_BL_1[MEM_BASE]
- PCIE2_RP_MEM_BL_1[MEM_LIMIT]
- PCIE2_RP_PRE_BL_1[PREFETCH_MEM_BASE]
- PCIE2_RP_PRE_BL_1[PREFETCH_MEM_LIMIT]

34.3.5 ELPG

The PCIE partition consists of both PCIE controllers. The PCIE partition can be put to power gating (ELPG) when both controllers have their links either in L2/3 or in disconnected states. The Tegra SOC is in running state when the PCIE partition is in ELPG.

1. Context Save/Restore

   In general, to ensure a controller correctly resume its operation after it exited power gating, software is required to perform context save and restore operations, where context is saved before the controller is put to ELPG and context is restored after the controller is bring out of ELPG. A controller's context can include register values, state machine states, and/or buffer statuses.

   Since all connected PCIE links are required to be put to L2/L3 for PCIE to be put to ELPG, software is not required to perform context save/restore for PCIE. This is because PCIE hierarchy reset is required to bring the links out of L2/L3, where software is required to re-enumerate the PCIE fabric. Thus for PCIE, SW should re-initialize AFI, PCIE Root Ports, and the PCIE hierarchy after exited ELPG or LP0.

2. PE_WAKE# in ELPG

   PCIE devices can assert PE_WAKE# to request the link to be bring to L0, which requires the PCIE partition to be powered. SW is notified of the assertion of PE_WAKE# by interrupt generated by the GPIO unit.

   GPIO interrupt generation to reflect PE_WAKE# assertion should be enabled before PCIE partition is put into ELPG by programming of the following register fields:

   GPIO_CNF_0.DD[BIT_3] to 'GPIO'

   GPIO_CNF_0.DD[LOCK_3] to 'DISABLE'

   GPIO_OE_0.DD[BIT_3] to 'TRI_STATE'

   GPIO_INT_LVL_0.DD[EDGE_3] to '0'

   GPIO_INT_LVL_0.DD[BIT_3] to 'LOW'

   GPIO_INT_ENB_0.DD[BIT_3] to 'ENABLE'

   After exiting ELPG, the PE_WAKE# interrupt should be disabled and the wake status should be cleared by programming the following register fields.

   GPIO_INT_ENB_0.DD[BIT_3] to 'DISABLE'

   GPIO_INT_CLR_0.DD[BIT_3] to 'CLEAR'

3. Power Gating Entry

   Partition power gating sequences consist of putting the lanes to IDDQ, asserting the resets to PCIE/AFI, disabling the clocks to PCIE/AFI, and remove the power to the partition, where power clamps are automatically enable by removing the power.

   IOPHY are put to IDDQ by setting the following register bits. Only lanes owned by PCIE should be programmed.

   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0] 1'b0
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1] 1'b0
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2] 1'b0
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3] 1'b0
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4] 1'b0

   AFI and PCIE resets are asserted by setting the following register bits:

   - CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_AFI_RST] 1'b1
   - CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_PCIE_RST] 1'b1
   - CLK_RST_CONTROLLER_RST_DEV_U_SET_0[SET_PCIEXCLK_RST] 1'b1

   AFI and PCIE clocks are enabled by setting the following register bits:

- CLK_RST_CONTROLLER_CLK_OUT_ENB_U_CLR_0[CLR_CLK_ENB_AFI] 1’b1
- CLK_RST_CONTROLLER_CLK_OUT_ENB_U_CLR_0[CLR_CLK_ENB_PCIE] 1’b1

AFI and PCIE power is removed by setting the following register fields:

- APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
- APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘PCX’

Then confirm the power removal by reading the following fields:

- APBDEV_PMC_PWRGATE_STATUS_0[PCX] equals ‘OFF’

4. Power Gating Exit

   Partition power ungating sequences consist of disable the power clamp to the partition, enabling the clocks to PCIE/ AFI, remove the power clamps, and deasserting the resets to PCIE/AFI.

   AFI and PCIE power is restored by setting the following register fields:

   - APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’
   - APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘PCX’

   Then confirm the power restore by reading the following fields:

   - APBDEV_PMC_PWRGATE_STATUS_0[PCX] equals ‘ON’

   AFI and PCIE clocks are enabled by setting the following register bits:

   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_SET_0[SET_CLK_ENB_AFI] 1’b1
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_SET_0[SET_CLK_ENB_PCIE] 1’b1

   AFI and PCIE power clamping is removed by setting the following register fields:

   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [PCX]

   Then confirm the power clamps removal by reading the following fields:

   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0 [PCX] equals ‘0’

   AFI and PCIE resets are deasserted by setting the following register bits:

   - CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_AFI_RST] 1’b1
   - CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_PCIE_RST] 1’b1

   IOPHY are brought out of IDDQ by setting the following register bits. Only lanes owned by PCIE should be programmed.

   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0] 1’b1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1] 1’b1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK2] 1’b1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK3] 1’b1
   - XUSB_PADCTL_USB3_PAD_MUX_0[FORCE_PCIE_PAD_IDDQ_DISABLE_MASK4] 1’b1

   PCIE XCLK domain reset is deasserted by setting the following CAR register bits:

   - CLK_RST_CONTROLLER_RST_DEV_U_CLR_0[CLR_PCIEXCLK_RST] 1’b1

34.3.6 LP0

PCIE partition must be put to ELPG before the system can enter LP0. PCIE partition can either be brought out of ELPG once the system exited LP0 or be kept under ELPG until the connected device signals wake event. Thus assertion of PE_WAKE# should be set as an LP0 wake event. The programming sequence of PCIE to support LP0 is identical to put PCIE in and out of ELPG.

- PE_WAKE# in LP0

The LP0 wake event triggered by the assertion PE_WAKE# should be enabled before the system goes into LP0 by programming of the following register fields:

- APBDEV_PMC_WAKE_LVL_0[EVENT[14]] to ‘ACTIVE_LOW’
- APBDEV_PMC_WAKE_MASK_0[EVENT[14]] to ‘ENABLE’

After exiting LP0, the PE_WAKE# wake event should be disabled and the wake status should be cleared by programming the following register fields.

- APBDEV_PMC_WAKE_MASK_0[EVENT[14]] to ‘DISABLE’
- APBDEV_PMC_WAKE_STATUS_0[EVENT[14]] to 1’b1

34.4 PCIe Registers

The PCI Express® registers are located as follows:

Table 216: PCIe Register Mapping

Name	Location	Description
NV_PCIE_AXI_RP_T0C0	NV_ADDRESS_MAP_PCIE_BASE	Start of PCIe extended config space for controller 0
NV_PCIE_AXI_RP_T0C1	(NV_PCIE_AXI_RP_T0C0) + 4096	Start of PCIe extended config space for controller 1
NV_PCIE_AXI_PCA_UFPCI_T0C0	(NV_PCIE_AXI_RP_T0C1) + 4096	Start of PCIe pca0 space
NV_PCIE_AXI_PCA_XCLK_T0C0	(NV_PCIE_AXI_PCA_UFPCI_T0C0) + 256)	Start of PCIe pca1 space
NV_PCIE_AXI_PADS	(NV_PCIE_AXI_RP_T0C1) + 8192	Start of PCIe pads space

34.4.1 PCI Compatible Configuration Registers

This section lists the Type 1 PCI Header registers as defined in PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

34.4.1.1 T_PCIE2_RP_DEV_ID

This register implements the Device ID and Vendor ID registers as defined by the PCI 2.0 Specification.

Device ID and Vendor ID Register

Offset: 0x00 | Read/Write: RO

Bit	Reset	Description
31:16	None	T_PCIE2_RP_DEV_ID_DEVICE_ID: The DEVICE_ID bits identify the particular device. This identifier is allocated by the vendor. NVIDIA uses unique Device IDs for Root Complexes with different Maximum Link Widths. FAEh: DEVICE_ID_TMS0_CTLR0_W4 FAFh: DEVICE_ID_TMS0_CTLR1_W1
15:0	10DEh	T_PCIE2_RP_DEV_ID_VENDOR_ID: The VENDOR_ID bits identify the manufacturer of the device. Valid vendor identifiers are allocated by the PCI SIG to ensure uniqueness. 10DEh: VENDOR_ID_NVIDIA (default)

34.4.1.2 T_PCIE2_RP_DEV_CTRL

Command and Status Register

Offset: 0x04 | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	T_PCIE2_RP_DEV_CTRL_DETECTED_PERR: This bit is set by the Primary side device whenever it receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Command register. 0h: DETECTED_PERR_NOT_ACTIVE (default) 1h: DETECTED_PERR_ACTIVE 1h: DETECTED_PERR_SET

Bit	R/W	Reset	Description
30	RW1C	0h	T_PCIE2_RP_DEV_CTRL_SIGNALED_SERR: This bit is set when the Primary side device sends an ERR_FATAL or ERR_NONFATAL message, and the SERR Enable bit is set. 0h: SIGNALED_SERR_NOT_ACTIVE (default) 1h: SIGNALED_SERR_ACTIVE 1h: SIGNALED_SERR_SET
29	RW1C	0h	T_PCIE2_RP_DEV_CTRL_RECEIVED_MASTER: This bit is set when the Primary side Requester receives a Completion with Unsupported Request Completion Status. 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_SET
28	RW1C	0h	T_PCIE2_RP_DEV_CTRL_RECEIVED_TARGET: This bit is set when the Primary side Requester receives a Completion with Completer Abort Completion Status. 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_SET
27	RW1C	0h	T_PCIE2_RP_DEV_CTRL_SIGNALED_TARGET: This bit is set when the Primary side device completes a Request using Completer Abort Completion Status. 0h: SIGNALED_TARGET_NO_ABORT (default) 1h: SIGNALED_TARGET_ABORT 1h: SIGNALED_TARGET_SET
26:25	R	0h	T_PCIE2_RP_DEV_CTRL_DEVSEL_TIMING: Does not apply to PCI Express. Hardwired to 0. 0h: DEVSEL_TIMING_FAST (default) 1h: DEVSEL_TIMING_MEDIUM 2h: DEVSEL_TIMING_SLOW
24	RW1C	0h	T_PCIE2_RP_DEV_CTRL_MASTER_DATA_PERR: Master Data Parity Error bit. This bit is set by the Primary side Requester if the Parity Error Response bit in the Command register is 1b and either of the following two conditions occurs: * Requester receives a Completion marked poisoned * Requester poisons a write Request If the Parity Error Response bit is 0b, this bit is never set. 0h: MASTER_DATA_PERR_NOT_ACTIVE (default) 1h: MASTER_DATA_PERR_ACTIVE 1h: MASTER_DATA_PERR_SET
23	R	0h	T_PCIE2_RP_DEV_CTRL_FAST_BACK2BACK: Does not apply to PCI Express. Hardwired to 0. 0h: FAST_BACK2BACK_INCAPABLE (default) 1h: FAST_BACK2BACK_CAPABLE
22	R	0	Reserved
21	R	0h	T_PCIE2_RP_DEV_CTRL_66MHZ: Does not apply to PCI Express. Hardwired to 0. 0h: 66MHZ_INCAPABLE (default) 1h: 66MHZ_CAPABLE
20	R	1h	T_PCIE2_RP_DEV_CTRL_CAPLIST: The CAPLIST bit indicates that the device configuration space includes a capabilities list. Hardwired to 1. 1h: CAPLIST_PRESENT (default) 0h: CAPLIST_NOT_PRESENT
19	R	0h	T_PCIE2_RP_DEV_CTRL_INTR_STATUS: The INTR_STATUS bit indicates that an INTx interrupt message is pending internally to the device. 0h: INTR_STATUS_NOT_ACTIVE (default) 1h: INTR_STATUS_ACTIVE
18:11	R	0	Reserved
10	R/W	0h	T_PCIE2_RP_DEV_CTRL_INTR_DISABLE: The INTR-DISABLE bit controls the ability of the device to generate INTx interrupt messages. When set, devices are prevented from generating INTx interrupt messages. 0h: INTR_DISABLE_INIT (default) 1h: INTR_DISABLE_YES 0h: INTR_DISABLE_NO

Bit	R/W	Reset	Description
9	R	0h	T_PCIE2_RP_DEV_CTRL_BACK2BACK: Does not apply to PCI Express. Hardwired to 0. 0h: BACK2BACK_DISABLED (default) 1h: BACK2BACK_ENABLED
8	R/W	0h	T_PCIE2_RP_DEV_CTRL_SERR: SERR Enable bit. This bit, when set, enables reporting of Non-fatal and Fatal errors detected by the Root Complex. In addition, this bit, when set, enables transmission by the primary interface of ERR_NONFATAL and ERR_FATAL error messages forwarded from the secondary interface. This bit does not affect the transmission of forwarded ERR_COR messages. 0h: SERR_DISABLED (default) 1h: SERR_ENABLED
7	R	0h	T_PCIE2_RP_DEV_CTRL_STEP: Does not apply to PCI Express. Hardwired to 0. 0h: STEP_DISABLED (default) 1h: STEP_ENABLED
6	R/W	0h	T_PCIE2_RP_DEV_CTRL_PERR: Parity Error Response bit. This bit, when set, controls the reporting of parity errors detected by the root complex (see Master Data Parity Error bit in the Status register). 0h: PERR_DISABLED (default) 1h: PERR_ENABLED
5	R	0h	T_PCIE2_RP_DEV_CTRL_PALETTE_SNOOP: Does not apply to PCI Express. Hardwired to 0. 0h: PALETTE_SNOOP_DISABLED (default) 1h: PALETTE_SNOOP_ENABLED
4	R	0h	T_PCIE2_RP_DEV_CTRL_WRITE_AND_INVAL: Does not apply to PCI Express. Hardwired to 0. 0h: WRITE_AND_INVAL_DISABLED (default) 1h: WRITE_AND_INVAL_ENABLED
3	R	0h	T_PCIE2_RP_DEV_CTRL_SPECIAL_CYCLE: Does not apply to PCI Express. Hardwired to 0. 0h: SPECIAL_CYCLE_DISABLED (default) 1h: SPECIAL_CYCLE_ENABLED
2	R/W	0h	T_PCIE2_RP_DEV_CTRL_BUS_MASTER: The BUS_MASTER bit controls the ability of the root complex to issue memory and I/O read/write requests. It controls forwarding of memory or I/O requests from the secondary interface to the primary interface. 0h: BUS_MASTER_DISABLED (default) 1h: BUS_MASTER_ENABLED
1	R/W	0h	T_PCIE2_RP_DEV_CTRL_MEMORY_SPACE: The MEMORY_SPACE bit indicates that the device will respond to memory space accesses on the primary interface. A value of 0 disables the device response on the primary interface. A value of 1 allows the device to forward memory transactions from the primary interface to the secondary interface. 0h: MEMORY_SPACE_DISABLED (default) 1h: MEMORY_SPACE_ENABLED
0	R/W	0h	T_PCIE2_RP_DEV_CTRL_IO_SPACE: The IO_SPACE bit indicates that the device will respond to I/O space accesses on the primary interface. A value of 0 disables the device response on the primary interface. A value of 1 allows forwarding of I/O accesses from the primary interface to the secondary interface. 0h: IO_SPACE_DISABLED (default) 1h: IO_SPACE_ENABLED

34.4.1.3 T_PCIE2_RP_REV_CC

This register implements the Revision ID and Class Code registers as defined by the PCI 2.0 Specification.

Revision ID and Class Code Registers

Offset: 0x08 | Read/Write: RO

Bit	Reset	Description
31:8	60400h	T_PCIE2_RP_REV_CC_CLASS_CODE: The CLASS_CODE bits identify the generic function of the device and (in some cases) a specific register-level programming interface. The register is broken into three byte-size fields. The upper byte (bits 31:24) is a base class code which broadly classifies the type of function the device performs. The middle-byte (bits 23:16) is a sub-class code which identifies more specifically the function of the device. The lower byte (bits 15:8) identifies a specific register-level programming interface, if any, so that device independent software can interact with the device. 60400h: CLASS_CODE_HOST (default) 60400h: CLASS_CODE_P2P
7:0	None	T_PCIE2_RP_REV_CC_REVISION_ID: The REVISION_ID bits specify a device specific revision identifier. The top nibble is the Major Revision and the bottom nibble is the Minor Revision. The Major Revision is changed every time there is an all layer change to the Controller (A, B, C ...). The Minor Revision is updated through metal edits each time there is a metal revision changing the functionality of this block. A1h: REVISION_ID_VAL

34.4.1.4 T_PCIE2_RP_MISC_1

This register implements the Cache Line Size, Latency Timer, Header Type, and BIST registers as defined by the PCI 2.0 Specification.

Cache Line Size Latency Timer Header Type and BIST Registers

Offset: 0x0c | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	T_PCIE2_RP_MISC_1_BIST: The BIST register field is optional and not used. Hardwired to 0. 0h: BIST_ZERO (default)
23	R	0h	T_PCIE2_RP_MISC_1_HEADER_TYPE1: The HEADER_TYPE bits identify the layout of the bytes 10h through 3Fh in configuration space and also whether or not the device contains multiple functions. Bit 23 in this register is used to identify a multi-function device. If the bit is 0, then the device is single function. If the bit is 1, then the device has multiple functions. 0h: HEADER_TYPE1_SINGLEFUNC (default) 1h: HEADER_TYPE1_MULTIFUNC
22:16	R	1h	T_PCIE2_RP_MISC_1_HEADER_TYPE0: The HEADER_TYPE bits identify the layout of the bytes 10h through 3Fh in configuration space and also whether or not the device contains multiple functions. Bits 22:16 in this register specify the layout of bytes 10h through 3Fh. PCI-Express Root Port Devices always employ Type 1 headers, hence this field is hardwired to 1h. 0h: HEADER_TYPE0_NON_BRIDGE 1h: HEADER_TYPE0_P2P_BRIDGE (default)
15:11	R	0h	T_PCIE2_RP_MISC_1_PLATENCY_TIMER: The primary/master latency timer does not apply to PCI Express. Hardwired to 0. 0h: PLATENCY_TIMER_0_CLOCKS (default)
10:8	R	0	Reserved
7:0	R/W	0h	T_PCIE2_RP_MISC_1_CACHE_LINE_SIZE: The cache line size register is set by the system firmware and the operating system to system cache line size. However, note that legacy PCI software may not always be able to program this field correctly especially in case of Hot-Plug devices. This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no impact on any PCI Express device functionality. 0h: CACHE_LINE_SIZE_0_BYTES (default)

34.4.1.5 T_PCIE2_RP_BAR_0

This register implements the Base Address Register 0 as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2. It is unused and hardwired to 0.

Base Address Register 0

Offset: 0x10 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_BAR_0_RESERVED: 0h: RESERVED_0 (default)

34.4.1.6 T_PCIE2_RP_BAR_1

This register implements the Base Address Register 1 as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2. It is unused and hardwired to 0.

Base Address Register 1

Offset: 0x14 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_BAR_1_RESERVED: 0h: RESERVED_0 (default)

34.4.1.7 T_PCIE2_RP_BN_LT

This register implements the Primary Bus Number, Secondary Bus Number, Subordinate Bus Number, and Secondary Latency Timer registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Primary Bus□Secondary Bus□Subordinate Bus Numbers□and Secondary Latency Timer Registers

Offset: 0x18 | Read/Write: R/W

Bit	R/W	Reset	Description
31:27	R	0h	T_PCIE2_RP_BN_LT_SLATENCY_TIMER: This field does not apply to PCI Express. Hardwired to 0. 0h: SLATENCY_TIMER_0_CLOCKS (default)
26:24	R	0	Reserved
23:16	R/W	0h	T_PCIE2_RP_BN_LT_SUB_BUS_NUMBER: The Subordinate Bus Number field is used to record the bus number of the highest numbered PCI bus segment which is behind the bridge. Configuration software programs the value in this field. The bridge uses this field in conjunction with the Secondary Bus Number field to determine when to respond to a Type 1 configuration transaction on the primary interface. 0h: SUB_BUS_NUMBER_0 (default) 1h: SUB_BUS_NUMBER_1 2h: SUB_BUS_NUMBER_2 FFh: SUB_BUS_NUMBER_255
15:8	R/W	0h	T_PCIE2_RP_BN_LT_SEC_BUS_NUMBER: The Secondary Bus Number field is used to record the bus number of the PCI bus segment to which the secondary interface of the bridge is connected. Configuration software programs the value in this field. The bridge uses this field to decode Type 1 configuration transactions on the primary interface and convert them to Type 0 accesses on the secondary interface. 0h: SEC_BUS_NUMBER_0 (default) 1h: SEC_BUS_NUMBER_1 2h: SEC_BUS_NUMBER_2 FFh: SEC_BUS_NUMBER_255
7:0	R/W	0h	T_PCIE2_RP_BN_LT_PRI_BUS_NUMBER: The Primary Bus Number field is used to record the bus number of the PCI bus segment to which the primary interface of the bridge is connected. Configuration software programs the value in this field. The bridge uses this field to decode Type 1 configuration transactions on the secondary interface that must be converted to Special Cycle transactions on the primary interface. 0h: PRI_BUS_NUMBER_0 (default)

34.4.1.8 T_PCIE2_RP_IO_BL_SS

This register implements the I/O Base, I/O Limit, and Secondary Status register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The I/O Base and I/O Limit fields define an address range that is used by the bridge to determine when to forward I/O transactions from one interface to the other.

The Secondary Status field is similar in function and bit definition to the Command and Status register defined above; however, its bits reflect status conditions of the secondary interface (the Command and Status register reflects the status conditions of the primary interface).

I/O Base I/O Limit and Secondary Status Register

Offset: 0x1c | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	T_PCIE2_RP_IO_BL_SS_DETECTED_PERR: This bit is set when the Secondary side device receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Bridge Control register. 0h: DETECTED_PERR_NOT_ACTIVE (default) 1h: DETECTED_PERR_ACTIVE 1h: DETECTED_PERR_SET
30	RW1C	0h	T_PCIE2_RP_IO_BL_SS_RECEIVED_SERR: This bit is set when the Secondary side device receives an ERR_FATAL or ERR_NONFATAL message. 0h: RECEIVED_SERR_NOT_ACTIVE (default) 1h: RECEIVED_SERR_ACTIVE 1h: RECEIVED_SERR_SET
29	RW1C	0h	T_PCIE2_RP_IO_BL_SS_RECEIVED_MASTER: This bit is set when the Secondary side device receives a Completion with Unsupported Request Completion Status. 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_SET
28	RW1C	0h	T_PCIE2_RP_IO_BL_SS_RECEIVED_TARGET: This bit is set when the Secondary side device receives a Completion with Completer Abort Completion Status. 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_SET
27	RW1C	0h	T_PCIE2_RP_IO_BL_SS_SIGNALED_TARGET: This bit is set when the Secondary side device completes a Request using Completer Abort Completion Status. 0h: SIGNALED_TARGET_NO_ABORT (default) 1h: SIGNALED_TARGET_ABORT 1h: SIGNALED_TARGET_SET
26:25	R	0h	T_PCIE2_RP_IO_BL_SS_DEVSEL_TIMING: Does not apply to PCI Express. Hardwired to 0. 0h: DEVSEL_TIMING_FAST (default) 1h: DEVSEL_TIMING_MEDIUM 2h: DEVSEL_TIMING_SLOW
24	RW1C	0h	T_PCIE2_RP_IO_BL_SS_MASTER_DATA_PERR: Master Data Parity Error bit. This bit is set by the Secondary side Requester if the Parity Error Response Enable bit in the Bridge Control register is 1b and either of the following two conditions occurs: * Requester receives a Completion marked poisoned * Requester poisons a write Request If the Parity Error Response bit is 0b, this bit is never set. 0h: MASTER_DATA_PERR_NOT_ACTIVE (default) 1h: MASTER_DATA_PERR_ACTIVE 1h: MASTER_DATA_PERR_SET
23	R	0h	T_PCIE2_RP_IO_BL_SS_FAST_BACK2BACK: Does not apply to PCI Express. Hardwired to 0. 0h: FAST_BACK2BACK_INCAPABLE (default) 1h: FAST_BACK2BACK_CAPABLE
22	R	0	Reserved
21	R	0h	T_PCIE2_RP_IO_BL_SS_66MHZ: Does not apply to PCI Express. Hardwired to 0. 0h: 66MHZ_INCAPABLE (default) 1h: 66MHZ_CAPABLE
20:16	R	0	Reserved

Bit	R/W	Reset	Description
15:12	R/W	0h	T_PCIE2_RP_IO_BL_SS_IO_LIMIT: This field corresponds to address bits AD[15:12] of the I/O limit. For the purpose of address decoding, the bridge assumes that the lower 12 address bits of the I/O limit are 0xFFF. The I/O Limit field can be programmed to a smaller value than the I/O Base register if there are no I/O addresses on the secondary side of the bridge. 0h: IO_LIMIT_ADDRESS_0 (default) 1h: IO_LIMIT_ADDRESS_256 2h: IO_LIMIT_ADDRESS_512 Fh: IO_LIMIT_ADDRESS_64K
11:8	R	1h	T_PCIE2_RP_IO_BL_SS_IO_LIMIT_SUPPORT: Identifies the I/O addressing support. When 0h, the device supports only 16-bit addressing. When 1h, it supports 32-bit addressing, in which case the I/O Limit Upper 16 Bits register is used to extend the I/O limit to 32 bits. Hardwired to 1h. 0h: IO_LIMIT_SUPPORT_16 1h: IO_LIMIT_SUPPORT_32 (default)
7:4	R/W	0h	T_PCIE2_RP_IO_BL_SS_IO_BASE: This field corresponds to address bits AD[15:12] of the I/O base address. For the purpose of address decoding, the bridge assumes that the lower 12 address bits, AD[11:0], of the I/O base address are zero. 0h: IO_BASE_ADDRESS_0 (default) 1h: IO_BASE_ADDRESS_256 2h: IO_BASE_ADDRESS_512 Fh: IO_BASE_ADDRESS_64K
3:0	R	1h	T_PCIE2_RP_IO_BL_SS_IO_BASE_SUPPORT: Identifies the I/O addressing support. When 0h, the device only supports 16-bit addressing. When 1h, it supports 32-bit addressing, in which case the I/O Base Upper 16 Bits register is used to extend the I/O base address to 32 bits. Hardwired to 1h. 0h: IO_BASE_SUPPORT_16 1h: IO_BASE_SUPPORT_32 (default)

34.4.1.9 T_PCIE2_RP_MEM_BL

This register implements the Memory Base and Memory Limit registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The Memory Base and Memory Limit fields define an address range that is used by the bridge to determine when to forward memory-mapped I/O transactions from one interface to the other.

Memory Base and Memory Limit Registers

Offset: 0x20 | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	T_PCIE2_RP_MEM_BL_MEM_LIMIT: This field corresponds to address bits AD[31:20] of the memory-mapped I/O limit. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the memory-mapped I/O limit are 0xFFFFF. The Memory Limit field must be programmed to a smaller value than the Memory Base field if there are no memory-mapped I/O addresses on the secondary side of the bridge. 0h: MEM_LIMIT_ADDRESS_0 (default) 1h: MEM_LIMIT_ADDRESS_1MEG 2h: MEM_LIMIT_ADDRESS_2MEG FFFh: MEM_LIMIT_ADDRESS_4GIG
19:16	R	0	Reserved
15:4	R/W	1h	T_PCIE2_RP_MEM_BL_MEM_BASE: This field corresponds to address bits AD[31:20] of the memory-mapped I/O base address. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the memory-mapped I/O base address are zero. 0h: MEM_BASE_ADDRESS_0 1h: MEM_BASE_ADDRESS_1MEG (default) 2h: MEM_BASE_ADDRESS_2MEG FFFh: MEM_BASE_ADDRESS_4GIG
3:0	R	0	Reserved

34.4.1.10 T_PCIE2_RP_PRE_BL

This register implements the Prefetchable Memory Base and Prefetchable Memory Limit registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The Prefetchable Memory Base and Prefetchable Memory Limit fields define an address range that is used by the bridge to determine when to forward prefetchable memory transactions from one interface to the other.

Prefetchable Memory Base and Prefetchable Memory Limit Registers

Offset: 0x24 | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	T_PCIE2_RP_PRE_BL_PREFETCH_MEM_LIMIT: This field corresponds to address bits AD[31:20] of the prefetchable memory limit. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the prefetchable memory limit are 0xFFFFF. The Prefetchable Memory Limit field must be programmed to a smaller value than the Prefetchable Memory Base field if there is no prefetchable memory on the secondary side of the bridge. 0h: PREFETCH_MEM_LIMIT_ADDRESS_0 (default) 1h: PREFETCH_MEM_LIMIT_ADDRESS_1MEG 2h: PREFETCH_MEM_LIMIT_ADDRESS_2MEG FFFh: PREFETCH_MEM_LIMIT_ADDRESS_4GIG
19:16	R	1h	T_PCIE2_RP_PRE_BL_L64BIT: Identifies the prefetchable memory addressing support. When 0h, the device supports only 32-bit addressing. When 1h, it supports 64-bit addressing, in which case the Prefetchable Limit Upper 32 Bits register is used to extend the prefetchable memory limit to 64 bits. Hardwired to 1h. 1h: L64BIT_YES (default)
15:4	R/W	1h	T_PCIE2_RP_PRE_BL_PREFETCH_MEM_BASE: This field corresponds to address bits AD[31:20] of the prefetchable memory base address. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the Prefetchable memory base address are zero. 0h: PREFETCH_MEM_BASE_ADDRESS_0 1h: PREFETCH_MEM_BASE_ADDRESS_1MEG (default) 2h: PREFETCH_MEM_BASE_ADDRESS_2MEG FFFh: PREFETCH_MEM_BASE_ADDRESS_4GIG
3:0	R	1h	T_PCIE2_RP_PRE_BL_B64BIT: Identifies the prefetchable memory addressing support. When 0h, the device supports only 32-bit addressing. When 1h, it supports 64-bit addressing, in which case the Prefetchable Base Upper 32 Bits register is used to extend the prefetchable memory base address to 64 bits. Hardwired to 1h. 1h: B64BIT_YES (default)

34.4.1.11 T_PCIE2_RP_PRE_BU32

This register implements the Prefetchable Memory Base Upper 32 Bits register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Prefetchable Memory Base Upper 32 Bits Register

Offset: 0x28 | Read/Write: R/W

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_PRE_BU32_BASE_UPPER_BITS: This register specifies the upper 32 bits, corresponding to AD[63:32], of the 64-bit prefetchable memory base address. 0h: BASE_UPPER_BITS_0 (default)

34.4.1.12 T_PCIE2_RP_PRE_LU32

This register implements the Prefetchable Memory Limit Upper 32 Bits register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Prefetchable Memory Limit Upper 32 Bits Register

Offset: 0x2c | Read/Write: R/W

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_PRE_LU32_LIMIT_UPPER_BITS: This register specifies the upper 32 bits, corresponding to AD[63:32], of the 64-bit prefetchable memory limit. 0h: LIMIT_UPPER_BITS_0 (default)

34.4.1.13 T_PCIE2_RP_IO_BL_U16

This register implements the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits Register

Offset: 0x30 | Read/Write: R/W

Bit	Reset	Description
31:16	0h	T_PCIE2_RP_IO_BL_U16_LIMIT_UPPER_BITS: This field specifies the upper 16 bits, corresponding to AD[31:16], of the 32-bit I/O limit. 0h: LIMIT_UPPER_BITS_0 (default)
15:0	0h	T_PCIE2_RP_IO_BL_U16_BASE_UPPER_BITS: This field specifies the upper 16 bits, corresponding to AD[31:16], of the 32-bit I/O base address. 0h: BASE_UPPER_BITS_0 (default)

34.4.1.14 T_PCIE2_RP_CAP_PTR

This register implements the Capabilities Pointer register as defined by the PCI 2.0 Specification.

Capabilities Pointer

Offset: 0x34 | Read/Write: RO

Bit	Reset	Description
31:8	0	Reserved
7:0	40h	T_PCIE2_RP_CAP_PTR_CAP_PTR: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. 40h: CAP_PTR_PM (default)

34.4.1.15 T_PCIE2_RP_ROM_BA

This register implements the optional Expansion ROM Base Address register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2. It is unused and hardwired to 0.

Expansion ROM Base Address Register

Offset: 0x38 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_ROM_BA_RESERVED: 0h: RESERVED_0 (default)

34.4.1.16 T_PCIE2_RP_INTR_BCR

This register implements the Interrupt Line, Interrupt Pin, and Bridge Control registers as defined by the PCI 2.0 Specification and PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Interrupt Line Interrupt Pin and Bridge Control Registers

Offset: 0x3c | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27	R	0h	T_PCIE2_RP_INTR_BCR_DIS_TIMER_SERR: Does not apply to PCI Express. Hardwired to 0. 0h: DIS_TIMER_SERR_DISABLED (default) 1h: DIS_TIMER_SERR_ENABLED
26	R	0h	T_PCIE2_RP_INTR_BCR_DIS_TIMER_STATUS: Does not apply to PCI Express. Hardwired to 0. 0h: DIS_TIMER_STATUS_NOT_ACTIVE (default) 1h: DIS_TIMER_STATUS_ACTIVE
25	R	0h	T_PCIE2_RP_INTR_BCR_SECONDARY_DIS_TIMER: Does not apply to PCI Express. Hardwired to 0. 0h: SECONDARY_DIS_TIMER_LONG (default) 1h: SECONDARY_DIS_TIMER_SHORT
24	R	0h	T_PCIE2_RP_INTR_BCR_PRIMARY_DIS_TIMER: Does not apply to PCI Express. Hardwired to 0. 0h: PRIMARY_DIS_TIMER_LONG (default) 1h: PRIMARY_DIS_TIMER_SHORT
23	R	0h	T_PCIE2_RP_INTR_BCR_FAST_B2B: Does not apply to PCI Express. Hardwired to 0. 0h: FAST_B2B_DISABLED (default) 1h: FAST_B2B_ENABLED
22	R/W	0h	T_PCIE2_RP_INTR_BCR_SB_RESET: Setting this bit triggers a hot reset on the corresponding PCI Express Port. 0h: SB_RESET_DISABLED (default) 1h: SB_RESET_ENABLED
21	R	0h	T_PCIE2_RP_INTR_BCR_MABORT: Does not apply to PCI Express. Hardwired to 0. 0h: MABORT_DISABLED (default) 1h: MABORT_ENABLED
20	R/W	0h	T_PCIE2_RP_INTR_BCR_VGA_16BITIO: When enabled, allows full 16-bit decode of I/O address range for VGA. 0h: VGA_16BITIO_DISABLED (default) 1h: VGA_16BITIO_ENABLED
19	R/W	0h	T_PCIE2_RP_INTR_BCR_VGA_ADDRESS: When enabled the P2P bridge will claim all of the legacy VGA addresses. Memory address range: 000A_0000 - 000B_FFFF I/O address ranges: 3B0-3BB, 3C0-3DF (including all aliases if VGA_16BITIO is not set). 0h: VGA_ADDRESS_DISABLED (default) 1h: VGA_ADDRESS_ENABLED
18	R/W	0h	T_PCIE2_RP_INTR_BCR_ISA_ADDRESS: Modifies the response by the bridge to ISA I/O addresses. This applies only to I/O addresses that are enabled by the I/O Base and I/O Limit registers and are in the first 64 KB of PCI I/O address space (0000 0000h to 0000 FFFFh). If this bit is set, the bridge will block any forwarding from primary to secondary of I/O transactions addressing the last 768 bytes in each 1-KB block. 0h: ISA_ADDRESS_DISABLED (default) 1h: ISA_ADDRESS_ENABLED
17	R/W	0h	T_PCIE2_RP_INTR_BCR_SERR_FORWARD: This bit controls forwarding of ERR_COR, ERR_NONFATAL and ERR_FATAL from secondary to primary. 0h: SERR_FORWARD_DISABLED (default) 1h: SERR_FORWARD_ENABLED
16	R/W	0h	T_PCIE2_RP_INTR_BCR_PERR_RESP: This bit controls the response to Poisoned TLPs. 0h: PERR_RESP_DISABLED (default) 1h: PERR_RESP_ENABLED

Bit	R/W	Reset	Description
15:8	R	1h	T_PCIE2_RP_INTR_BCR_INTR_PIN: This field contains the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that do not use an interrupt pin must put a 0 in this field. 0h: INTR_PIN_NONE 1h: INTR_PIN_INTA (default) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	R/W	0h	T_PCIE2_RP_INTR_BCR_INTR_LINE: This field contains the interrupt routing information. The field must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this field as it initializes and configures the system. The value in this field tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is written to 0xff (no connection) by software if the bridge does not use an interrupt pin. Some PCI BIOSes cannot handle aliased INTR_LINEs. Some PCI BIOSes cannot handle INTR_LINE initialized to 0xff. 0h: INTR_LINE_IRQ0 (default) 1h: INTR_LINE_IRQ1 Fh: INTR_LINE_IRQ15 FFh: INTR_LINE_UNKNOWN

34.4.2 PCI Subsystem ID and Subsystem Vendor ID Capability Registers

This section lists the PCI Subsystem ID and Subsystem Vendor ID Capabilities List registers as defined in PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

34.4.2.1 T_PCIE2_RP_SS_0

This register implements the first register of the Subsystem ID and Subsystem Vendor ID Capability list item, as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Subsystem ID and Subsystem Vendor ID Capability Register 0

Offset: 0x40 | Read/Write: RO

Bit	Reset	Description
31:16	0	Reserved
15:8	48h	T_PCIE2_RP_SS_0_NEXT_PTR: This read-only field points to the next capabilities list item. 48h: NEXT_PTR_PM (default)
7:0	Dh	T_PCIE2_RP_SS_0_CAP_ID: This read-only field is used to detect the presence of the SSID/SSVID registers in a PCI-to-PCI bridge. Hardwired to 0Dh. Dh: CAP_ID_SS (default)

34.4.2.2 T_PCIE2_RP_SS_1

This register implements the second register of the.Subsystem ID and Subsystem Vendor ID Capability list item, as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Subsystem ID and Subsystem Vendor ID Capability Register 1

Offset: 0x44 | Read/Write: RO

Bit	Reset	Description
31:16	0h	T_PCIE2_RP_SS_1_SSID: The SSID identifies the particular add-in card or subsystem and is assigned by the vendor. 0h: SSID_INIT (default)
15:0	10DEh	T_PCIE2_RP_SS_1_SSVID: The SSVID identifies the manufacturer of the add-in card or subsystem. The SSVID is assigned by PCI-SIG to ensure uniqueness (the Vendor ID is used as the SSVID also). 10DEh: SSVID_INIT (default)

34.4.3 PCI Power Management Capability Structure Registers

This section lists the PCI Power Management Capabilities List registers as defined in PCI Express Base Specification, Revision 3.0.

34.4.3.1 T_PCIE2_RP_PM_0

This register implements the Power Management Capabilities register as defined in Section 7.6 of PCI Express Specifications.

Power Management Capabilities Register

Offset: 0x48 | Read/Write: RO

Bit	Reset	Description
31:27	1Fh	T_PCIE2_RP_PM_0_PME_SUPPORT: This 5-bit field indicates the power states in which the function may assert PME#. A value of 0b for any bit indicates that the function is not capable of asserting the PME# signal while in that power state. bit(11) XXXX1b = PME# can be asserted from D0 bit(12) XXX1Xb = PME# can be asserted from D1 bit(13) XX1XXb = PME# can be asserted from D2 bit(14) X1XXXb = PME# can be asserted from D3hot bit(15) 1XXXXb = PME# can be asserted from D3cold Bits 31, 30, and 27 must be set to 1b for PCI-PCI Bridge structures representing Ports on Root Complexes/ Switches to indicate that the Bridge will forward PME Messages. 1Fh: PME_SUPPORT_YES (default) 0h: PME_SUPPORT_NO
26	0h	T_PCIE2_RP_PM_0_D2_SUPPORT: If this bit is a "1", this function supports the D2 Power Management State. 1h: D2_SUPPORT_YES 0h: D2_SUPPORT_NO (default)
25	0h	T_PCIE2_RP_PM_0_D1_SUPPORT: If this bit is a "1", this function supports the D1 Power Management State. 1h: D1_SUPPORT_YES 0h: D1_SUPPORT_NO (default)
24:22	0h	T_PCIE2_RP_PM_0_AUX_CURRENT: This 3 bit field reports the 3.3Vaux auxiliary current requirements for the PCI function. Bit 3.3Vaux 8 7 6 Max. Current Required 1 1 1 375 mA 1 1 0 320 mA 1 0 1 270 mA 1 0 0 220 mA 0 1 1 160 mA 0 1 0 100 mA 0 0 1 55 mA 0 0 0 0 (self powered) 0h: AUX_CURRENT_0 (default)
21	0h	T_PCIE2_RP_PM_0_DEV_SPEC_INIT: The Device Specific Initialization bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. Note that this bit is not used by some operating systems. Microsoft Windows and Windows NT, for instance, do not use this bit to determine whether to use D3. Instead, they use the driver's capabilities to determine this. A "1" indicates that the function requires a device specific initialization sequence following transition to the D0 uninitialized state. 0h: DEV_SPEC_INIT_NOT_NEEDED (default) 1h: DEV_SPEC_INIT_NEEDED
20	0	Reserved
19	0h	T_PCIE2_RP_PM_0_PME_CLOCK: Does not apply to PCI Express. Must be hardwired to 0. 0h: PME_CLOCK_NOT_NEEDED (default) 1h: PME_CLOCK_NEEDED

Bit	Reset	Description
18:16	3h	T_PCIE2_RP_PM_0_PCIPM_REV: A value of 010b indicates that this function complies with Revision 1.1 of the PCI Power Management Interface Specification. 3h: PCIPM_REV_12 (default) 2h: PCIPM_REV_11
15:8	50h	T_PCIE2_RP_PM_0_NEXT_PTR: This read-only field points to the next capabilities list item. 50h: NEXT_PTR_MSI (default)
7:0	1h	T_PCIE2_RP_PM_0_CAP_ID: This read-only field is used to detect the presence of the Power Management Capability registers in a PCI-to-PCI bridge. Hardwired to 01h. 1h: CAP_ID_PM (default)

34.4.3.2 T_PCIE2_RP_PM_1

This register implements the Power Management Status/Control register as defined in Section 7.6 of PCI Express Specification.

Power Management Status/Control Register

Offset: 0x4c | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	T_PCIE2_RP_PM_1_PME_DATA: The PME Data register is not implemented. Hardwired to 0. 0h: PME_DATA_UNS (default)
23	R	0h	T_PCIE2_RP_PM_1_PME_BPCC: The bus power/clock control mechanism is not used. Hardwired to 0. 0h: PME_BPCC_UNS (default)
22	R	0h	T_PCIE2_RP_PM_1_PME_B2B3: The bus power/clock control mechanism is not used. Therefore the B2/B3 support bit is hardwired to 0. 0h: PME_B2B3_UNS (default)
21:16	R	0	Reserved
15	RW1C	0h	T_PCIE2_RP_PM_1_PME_STATUS: This bit is set when the function would normally assert the PME# signal independent of the state of the PME_En bit. Writing a "1" to this bit will clear it and cause the function to stop asserting a PME# (if enabled). Writing a "0" has no effect. This bit is sticky and must be explicitly cleared by the operating system each time the operating system is initially loaded. NOTE: This field is reset by Cold Reset 0h: PME_STATUS_NOT_ACTIVE (default) 1h: PME_STATUS_ACTIVE 1h: PME_STATUS_SET
14:13	R	0h	T_PCIE2_RP_PM_1_PME_DATA_SCALE: The PME Data register is not implemented. Hardwired to 0. 0h: PME_DATA_SCALE_UNS (default)
12:9	R	0h	T_PCIE2_RP_PM_1_PME_DATA_SEL: The PME Data register is not implemented. Hardwired to 0. 0h: PME_DATA_SEL_UNS (default)
8	R/W	0h	T_PCIE2_RP_PM_1_PME: A "1" enables the function to assert PME#. When "0", PME# assertion is disabled. This bit is sticky and must be explicitly cleared by the operating system each time it is initially loaded. NOTE: This field is reset by Cold Reset 0h: PME_DISABLE (default) 1h: PME_ENABLE
7:2	R	0	Reserved

Bit	R/W	Reset	Description
1:0	R/W	0h	T_PCIE2_RP_PM_1_PWR_STATE: This 2-bit field is used both to determine the current power state of a function and to set the function into a new power state. The definition of the field values is given below. 00b - D0 01b - D1 10b - D2 11b - D3hot If software attempts to write an unsupported, optional state to this field, the write operation must complete normally on the bus; however, the data is discarded and no state change occurs. 0h: PWR_STATE_D0 (default) 1h: PWR_STATE_D1 2h: PWR_STATE_D2 3h: PWR_STATE_D3HOT

34.4.4 PCI MSI Capability Structure Registers

This section lists the PCI MSI Capabilities List registers as defined in PCI Local Bus Specification, Revision 2.0.

34.4.4.1 T_PCIE2_RP_MSI_CTRL

The MSI capability structure is required for all PCI Express devices that are capable of generating interrupts.

MSI enables a device to request service by writing a system-specified message to a system-specified address. The transaction address specifies the message destination and the transaction data specifies the message. System software initializes the message destination and message during device configuration.

MSI Control and Capability Registers

Offset: 0x50 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	T_PCIE2_RP_MSI_CTRL_RSVD: The RSVD field is reserved by the specification. 0h: RSVD_0 (default)
23	R	1h	T_PCIE2_RP_MSI_CTRL_64BIT_CAP: The 64BIT_CAP field indicates if the function is capable of generating a 64-bit message address. If 1, the function is capable. 1h: 64BIT_CAP_TRUE (default)
22:20	R/W	0h	T_PCIE2_RP_MSI_CTRL_MULT_EN: The MULT_EN field indicates the number of allocated messages. It is always aligned to a power of 2. 0h: MULT_EN_CODE0 (default) 1h: MULT_EN_CODE2 2h: MULT_EN_CODE4 3h: MULT_EN_CODE8
19:17	R	1h	T_PCIE2_RP_MSI_CTRL_MULT_CAP: The MULT_CAP field determines the number of requested messages. It should always be a power of 2. 1h: MULT_CAP_CODE2 (default)
16	R/W	0h	T_PCIE2_RP_MSI_CTRL_MSI: The MSI bit controls if the function is permitted to use message signaled interrupt to request service. If 1, the function is permitted to use MSI and prohibits the use of INTx messages. 0h: MSI_DISABLE (default) 1h: MSI_ENABLE
15:8	R	None	T_PCIE2_RP_MSI_CTRL_NEXT_PTR: The NEXT_PTR field points to the next item in the capabilities list. 60h: NEXT_PTR_MSIMAP 80h: NEXT_PTR_PCIEXP
7:0	R	5h	T_PCIE2_RP_MSI_CTRL_CAP_ID: This read-only field is used to detect the presence of the Message Signaled Interrupt Capability registers in a PCI-to-PCI bridge. Hardwired to 05h. 5h: CAP_ID_MSI (default)

34.4.4.2 T_PCIE2_RP_MSI_LOW_ADDR

The contents of this register specify the lower 32 bits of the Dword aligned address for the MSI memory write transaction.

MSI Message Lower Address Register

Offset: 0x54 | Read/Write: R/W

Bit	R/W	Reset	Description
31:2	R/W	0h	T_PCIE2_RP_MSI_LOW_ADDR_DWORD: Bits 31:2 of the address. 0h: DWORD_0 (default)
1:0	R	0h	T_PCIE2_RP_MSI_LOW_ADDR_RSVD: The address is always DWORD aligned, hence bits 1:0 of this register are hardwired to 0. 0h: RSVD_0 (default)

34.4.4.3 T_PCIE2_RP_MSI_UPPER_ADDR

The contents of this register specify the upper 32 bits of the 64-bit MSI message address.

MSI Message Upper Address Register

Offset: 0x58 | Read/Write: R/W

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_MSI_UPPER_ADDR_DWORD: Bits 63:32 of the address. 0h: DWORD_0 (default)

34.4.4.4 T_PCIE2_RP_MSI_DATA

The contents of this register specify the data that is written to the MSI message address.

MSI Message Data Register

Offset: 0x5c | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0h	T_PCIE2_RP_MSI_DATA_RSVD: 0h: RSVD_0 (default)
15:0	R/W	0h	T_PCIE2_RP_MSI_DATA_DATA: The MSI message data. 0h: DATA_0 (default)

34.4.5 PCI Express Capability Structure Registers

This section list the PCI Express Capability List registers as defined in PCI Express Base Specification, Revision 3.0.

34.4.5.1 T_PCIE2_RP_PCI_EXPRESS_CAPABILITY

The PCI Express Capability List register enumerates the PCI Express Capability Structure in PCI 2.3 configuration space capability list. It also identifies PCI Express device type and associated capabilities.

PCI Express Capability List Register

Offset: 0x80 | Read/Write: RO

Bit	Reset	Description
31:30	0	Reserved

Bit	Reset	Description
29:25	0h	T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_INTERRUPT_MESSAGE_NUMBER: If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register of this capability structure are set. Hardware is required to update this field so that it is correct if the number of the MSI Messages assigned to the device changes. 0h: INTERRUPT_MESSAGE_NUMBER_ZERO (default)
24	1h	T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_SLOT_IMPLEMENTED: This bit when set indicates that the PCI Express Link associated with this port is connected to a slot (as compared to being connected to an integrated component or being disabled). This field is valid for the following PCI Express device/Port Types: * Root Port of PCI Express Root Complex. * Downstream Port of PCI Express Switch. 1h: SLOT_IMPLEMENTED_INIT (default)
23:20	4h	T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_DEVICE_PORT_TYPE: Indicates the type of PCI Express logical device. Defined encodings are: 0000b = PCI Express Endpoint device. 0001b = Legacy PCI Express Endpoint device. 0100b = Root Port of PCI Express Root Complex. 0101b = Upstream Port of PCI Express Switch. 0110b = Downstream Port of PCI Express Switch. 0111b = PCI Express-to-PCI/PCI-X Bridge. 1000b = PCI/PCI-X to PCI Express Bridge 4h: DEVICE_PORT_TYPE_INIT (default)
19:16	2h	T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_VERSION: Indicates PCI_SIG defined PCI Express capability structure version number. Must be 1h for versions 1.0a and 1.1 of the PCI-Express Specification. Must be 2h for devices compliant to the Express Capabilities Register Expansion ECN. 2h: VERSION_INIT (default) 1h: VERSION_1 2h: VERSION_2
15:8	0h	T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_LIST_NEXT_CAPABILITY_PTR: The offset to the next PCI capability structure or 0h if no other items exist in the linked list of capabilities. 0h: LIST_NEXT_CAPABILITY_PTR_INIT (default)
7:0	10h	T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_LIST_CAPABILITY_ID: Indicates PCI Express Capability Structure. This field must return a Capability ID of 10h indicating that this is a PCI Express Capability Structure. 10h: LIST_CAPABILITY_ID_INIT (default)

34.4.5.2 T_PCIE2_RP_DEVICE_CAPABILITY

The Device Capabilities register identifies PCI Express device specific capabilities.

Device Capabilities Register

Offset: 0x84 | Read/Write: RO

Bit	Reset	Description
31:28	0	Reserved
27:26	0h	T_PCIE2_RP_DEVICE_CAPABILITY_CAPTURED_SLOT_POWER_LIMIT_SCALE: Specifies the scale used for the Slot Power Limit Value. Range of Values: 00b = 1.0x 01b = 0.1x 11b = 0.01x 11b = 0.001x This value is set by the Set_Slot_Power_Limit message or hardwired to 00b. The default value is all 00b. 0h: CAPTURED_SLOT_POWER_LIMIT_SCALE_INIT (default)
25:18	0h	T_PCIE2_RP_DEVICE_CAPABILITY_CAPTURED_SLOT_POWER_LIMIT_VALUE: In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This value is set by the Set_Slot_Power_Limit message or hardwired to 0000_0000b. The default value is 0000_0000b. 0h: CAPTURED_SLOT_POWER_LIMIT_VALUE_INIT (default)

Bit	Reset	Description
17:16	0	Reserved
15	1h	T_PCIE2_RP_DEVICE_CAPABILITY_ROLE_BASED_ERR_REPORTING: This bit, when set, indicates that the device implements the functionality originally defined in the Error Reporting ECN for PCI Express Base Specification 1.0a and later incorporated into PCI Express Base Specification 1.1. 1h: ROLE_BASED_ERR_REPORTING_INIT (default)
14	0h	T_PCIE2_RP_DEVICE_CAPABILITY_POWER_INDICATOR_PRESENT: This bit, when set, indicates that a Power Indicator is implemented on the card or module. 0h: POWER_INDICATOR_PRESENT_INIT (default)
13	0h	T_PCIE2_RP_DEVICE_CAPABILITY_ATTENTION_INDICATOR_PRESENT: This bit, when set, indicates that an Attention Indicator is implemented on the card or module. 0h: ATTENTION_INDICATOR_PRESENT_INIT (default)
12	0h	T_PCIE2_RP_DEVICE_CAPABILITY_ATTENTION_BUTTON_PRESENT: This bit when set indicates that an Attention Button is implemented on the card or module. 0h: ATTENTION_BUTTON_PRESENT_INIT (default)
11:9	0h	T_PCIE2_RP_DEVICE_CAPABILITY_ENDPOINT_L1_ACCEPTABLE_LATENCY: This field indicates acceptable latency that an Endpoint can withstand due to the transition from L1 state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. 000b -- Less than 1 µs. 001b -- 1 µs to less than 2 µs. 010b -- 2 µs to less than 4 µs. 011b -- 4 µs to less than 8 us. 100b – 8 µs to less than 16 µs. 101b -- 16 µs to less than 32 µs. 110b -- 32 µs-64 µs. 111b -- More than 64 µs. 0h: ENDPOINT_L1_ACCEPTABLE_LATENCY_INIT (default)
8:6	0h	T_PCIE2_RP_DEVICE_CAPABILITY_ENDPOINT_L0S_ACCEPTABLE_LATENCY: This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. 000b -- Less than 64 ns. 001b -- 64 ns to less than 128 ns. 010b -- 128 ns to less than 256 ns. 011b -- 256 ns to less than 512 ns. 100b -- 512 ns to less than 1 µs. 101b -- 1 µs to less than 2 µs. 110b -- 2 µs-4µs. 111b -- More than 4 µs. 0h: ENDPOINT_L0S_ACCEPTABLE_LATENCY_INIT (default)
5	None	T_PCIE2_RP_DEVICE_CAPABILITY_EXTENDED_TAG_FIELD_SIZE: This field indicates the maximum supported size of the Tag field. Defined encodings are: 0b = 5-bit Tag field supported. 1b = 8-bit Tag field supported.
4:3	0h	T_PCIE2_RP_DEVICE_CAPABILITY_PHANTOM_FUNCTIONS_SUPPORTED: This field indicates the support for use of unclaimed function numbers to extend the number of outstanding transactions allowed by logically combining unclaimed function numbers (called Phantom Functions) with the Tag identifier. This field indicates the number of most significant bits of the function number portion of Requester ID that are logically combined with the Tag identifier. Defined encodings are: 00b -- No function number bits used for Phantom Functions; device may implement all function numbers. 01b -- First most significant bit of the function number in Requester ID used for Phantom Functions; device may implement functions 0-3. Functions 0, 1, 2 and 3 may claim functions 4, 5, 6, and 7 as Phantom Functions respectively. 10b -- First two most significant bits of the function number in Requester ID used for Phantom Functions; device may implement functions 0-1. Function 0 may claim functions 2, 4 and 6 as Phantom Functions, function 1 may claim functions 3, 5 and 7 as Phantom Functions. 11b -- All three bits of function number in Requester ID used for Phantom Functions; device must be a single function 0 device that may claim all other functions as Phantom Functions. Note: The Phantom Function support for the device must be enabled by the corresponding control field in the Device Control Register. 0h: PHANTOM_FUNCTIONS_SUPPORTED_INIT (default)

Bit	Reset	Description
2:0	0h	T_PCIE2_RP_DEVICE_CAPABILITY_MAX_PAYLOAD_SIZE: This field indicates the maximum payload size that the device can support for TLP's. 000b --128B maximum payload size 001b -- 256B maximum payload size 010b -- 512B maximum payload size 011b -- 1024B maximum payload size 100b -- 2048B maximum payload size 101b -- 4096B maximum payload size 110b -- Reserved 111b -- Reserved 0h: MAX_PAYLOAD_SIZE_INIT (default)

34.4.5.3 T_PCIE2_RP_DEVICE_CONTROL_STATUS

The Device Control register controls PCI Express device specific parameters. It also provides information about PCI Express device specific parameters.

Device Control and Device Status Registers

Offset: 0x88 | Read/Write: R/W

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21	R	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_TRANSACTIONS_PENDING: This bit when set indicates that a device has issued Non Posted requests which have not been completed. A device reports this bit cleared only when all Completions for any outstanding Non-Posted Requests have been received. 0h: TRANSACTIONS_PENDING_INIT (default)
20	R	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_AUX_POWER_DETECTED: Devices that require AUX power report this bit as set if AUX power is detected by the device. 0h: AUX_POWER_DETECTED_INIT (default)
19	RW1C	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_UNSUPP_REQUEST_DETECTED: This bit indicates that the device received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register. NOTE: This field is reset by Cold Reset 0h: UNSUPP_REQUEST_DETECTED_INIT (default) 1h: UNSUPP_REQUEST_DETECTED_SET
18	RW1C	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_FATAL_ERROR_DETECTED: This bit indicates status of fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. 0h: FATAL_ERROR_DETECTED_INIT (default) 1h: FATAL_ERROR_DETECTED_SET
17	RW1C	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_NON_FATAL_ERROR_DETECTED: This bit indicates status of non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. NOTE: This field is reset by Cold Reset 0h: NON_FATAL_ERROR_DETECTED_INIT (default) 1h: NON_FATAL_ERROR_DETECTED_SET
16	RW1C	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_CORR_ERROR_DETECTED: This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. Default value of this field is 0. NOTE: This field is reset by Cold Reset 0h: CORR_ERROR_DETECTED_INIT (default) 1h: CORR_ERROR_DETECTED_SET
15	R	0	Reserved

Bit	R/W	Reset	Description
14:12	R/W	2h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_MAX_READ_REQUEST_SIZE: This field sets the maximum Read Request size for the Device as a Requester. The Device must not generate read requests with size exceeding the set value. Defined encodings for this field are: 000b -- 128B maximum payload size 001b -- 256B maximum payload size 010b -- 512B maximum payload size 011b -- 1024B maximum payload size 100b -- 2048B maximum payload size 101b -- 4096B maximum payload size 110b -- Reserved 111b – Reserved Devices that do not generate Read Requests larger than 128 bytes are permitted to implement this field as Read Only (RO) with: a value of 000b. 2h: MAX_READ_REQUEST_SIZE_INIT (default)
11	R/W	1h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_ENABLE_NO_SNOOP: If this bit is set to 1, the device is permitted to set the No Snoop Bit in the Requester Attributes of transactions it initiates that do not require hardware enforced cache coherency. Even when this bit is set to 1, a device may only set the No Snoop attribute on a transaction when it can guarantee that the address of the transaction is not stored in any cache in the system. This bit may be hardwired to 0 if a device never sets the No Snoop attribute in transactions it initiates. 1h: ENABLE_NO_SNOOP_INIT (default)
10	R/W	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_AUXILLARY_POWER_PM_ENABLE: This bit when set enables a device to draw AUX power independent of PME AUX power. AUX power is allocated as requested in the AUX_Current field of the Power Management Capabilities Register (PMC), independent of the PME_En bit in the Power Management Control/Status Register (PMCSR). For multi-function devices, a component is allowed to draw AUX power if at least one of the functions has this bit set. Devices that do not implement this capability hardwire this bit to 0. NOTE: This field is reset by Cold Reset 0h: AUXILLARY_POWER_PM_ENABLE_INIT (default)
9	R	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_PHANTOM_FUNCTIONS_ENABLE: When set, this bit enables a device to use unclaimed functions as Phantom Functions to extend the number of outstanding transaction identifiers. If the bit is cleared, the device is not allowed to use Phantom Functions. Devices that do not implement this capability hardwire this bit to 0. 0h: PHANTOM_FUNCTIONS_ENABLE_INIT (default)
8	R/W	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_EXTENDED_TAG_FIELD_ENABLE: When set, this bit enables a device to use an 8-bit Tag field as a requester. If the bit is cleared, the device is restricted to a 5-bit Tag field. Devices that do not implement this capability hardwire this bit to 0. 0h: EXTENDED_TAG_FIELD_ENABLE_INIT (default)
7:5	R/W	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_MAX_PAYLOAD_SIZE: This field sets the maximum TLP payload size for the device. As a receiver, the device must handle TLP's as large as the set value; as transmitter, the device must not generate TLPs exceeding the set value. Permissible values that can be programmed are indicated by the Max_Payload_Size supported in the Device Capabilities register. Defined encodings for this field are: • 000b -- 128B maximum payload size • 001b -- 256B maximum payload size • 010b -- 512B maximum payload size • 011b -- 1024B maximum payload size • 100b -- 2048B maximum payload size • 101b -- 4096B maximum payload size • 110b -- Reserved • 111b -- Reserved 0h: MAX_PAYLOAD_SIZE_INIT (default)
4	R/W	1h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_ENABLE_RELAXED_ORDERING: If this bit is set, the device is permitted to set the Relaxed Ordering bit in the Attributes field of transactions it initiates that do not require strong write ordering. This bit may be hardwired to 0 if a device never sets the Relaxed Ordering Attribute in transactions it initiates as a requester. 1h: ENABLE_RELAXED_ORDERING_INIT (default)

Bit	R/W	Reset	Description
3	R/W	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_UNSUPP_REQ_REPORTING_ENABLE: This bit enables reporting of Unsupported Requests when set. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. Note: The reporting of error messages (ERR_COR, ERR_NONFATAl, ERR_FATAL) received by Root Port is controlled exclusively by Root Control Register. 0h: UNSUPP_REQ_REPORTING_ENABLE_INIT (default)
2	R/W	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_FATAL_ERROR_REPORTING_ENABLE: This bit controls reporting of fatal errors. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. 0h: FATAL_ERROR_REPORTING_ENABLE_INIT (default)
1	R/W	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_NON_FATAL_ERROR_REPORTING_ENABLE: This bit controls reporting of non-fatal errors. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. 0h: NON_FATAL_ERROR_REPORTING_ENABLE_INIT (default)
0	R/W	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_CORR_ERROR_REPORTING_ENABLE: This bit controls reporting of correctable errors. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. 0h: CORR_ERROR_REPORTING_ENABLE_INIT (default)

34.4.5.4 T_PCIE2_RP_LINK_CAPABILITIES

The Link Capabilities Register identifies PCI Express Link specific capabilities.

Link Capabilities Register

Offset: 0x8c | Read/Write: RO

Bit	Reset	Description
31:24	None	T_PCIE2_RP_LINK_CAPABILITIES_PORT_NUMBER: This field indicates the PCI Express port number for the given PCI Express Link.
23:22	None	T_PCIE2_RP_LINK_CAPABILITIES_RESERVED: Reserved.
21	0h	T_PCIE2_RP_LINK_BW_NOTIFY: 0h: INIT (default)
20	1h	T_PCIE2_RP_LINK_CAPABILITIES_LINKACTV_REPORTING: For a Downstream Port, this bit must be set to 1b if the component supports the optional capability of reporting the DL_Active state of the Data Link Control and Management State Machine. For a hot-plug capable Downstream Port (as indicated by the Hot-Plug Capable field of the Slot Capabilities register), this bit must be set to 1b. 1h: LINKACTV_REPORTING_INIT (default)
19	0h	T_PCIE2_RP_LINK_CAPABILITIES_SURPRISE_DOWN_ERPT_CAP: For a Downstream Port, this bit must be set to 1b if the component supports the optional capability of detecting and reporting a Surprise Down error condition. 0h: SURPRISE_DOWN_ERPT_CAP_INIT (default)
18	0h	T_PCIE2_RP_LINK_CAPABILITIES_CLOCK_PM: 0h: CLOCK_PM_INIT (default)
17:15	2h	T_PCIE2_RP_LINK_CAPABILITIES_L1_EXIT_LATENCY: This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L1 to L0. Defined encodings are: 000b -- Less than 1 µs. 001b -- 1 µs to less than 2 µs. 010b -- 2 µs to less than 4 µs. 011b -- 4 µs to less than 8 µs. 100b -- 8 µs to less than 16 µs. 101b -- 16 µs to less than 32 µs. 110b -- 32 µs-64 µs. 111b -- More than 64 us. 2h: L1_EXIT_LATENCY_INIT (default)

Bit	Reset	Description
14:12	3h	T_PCIE2_RP_LINK_CAPABILITIES_L0S_EXIT_LATENCY: This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L0s state to L0. Defined encodings are: 000b -- Less than 64 ns. 001b -- 64 ns to less than 128 ns. 010b -- 128 ns to less than 256 ns. 011b -- 256 ns to less than 512 ns. 100b -- 512 ns to less than 1 µs. 101b -- 1 µs to less than 2 µs. 110b -- 2 µs-4 µs. 111b -- Reserved. 3h: L0S_EXIT_LATENCY_INIT (default)
11:10	11h	T_PCIE2_RP_LINK_CAPABILITIES_ACTIVE_STATE_LINK_PM_SUPPORT: This field indicates the level of active state power management supported on the given PCI Express Link. Defined encodings are: 00b -- Reserved 01b -- L0s Entry Supported 10b -- Reserved 11b -- L0s and L1 Supported 11h: ACTIVE_STATE_LINK_PM_SUPPORT_INIT (default)
9:4	None	T_PCIE2_RP_LINK_CAPABILITIES_MAX_LINK_WIDTH: This field indicates the maximum width of the given PCI Express Link. Defined encodings are: 000000b -- Reserved 000001b -- x1 000010b -- x2 000100b -- x4 001000b -- x8 001100b -- x12 010000b -- x16 100000b -- x32
3:0	2h	T_PCIE2_RP_LINK_CAPABILITIES_MAX_LINK_SPEED: This field indicates the maximum link speed of the given PCI Express Link. Defined encodings are: 0001b = 2.5 Gb/s Link All other encodings are reserved. 2h: MAX_LINK_SPEED_INIT (default)

34.4.5.5 T_PCIE2_RP_LINK_CONTROL_STATUS

The Link Control register controls PCI Express Link specific parameters. It also provides information about PCI Express Link specific parameters.

Link Control and Link Status Registers

Offset: 0x90 | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_AUTO_BANDWIDTH
30	RW1C	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_BW_MANAGEMENT
29	R	None	T_PCIE2_RP_LINK_CONTROL_STATUS_DL_LINK_ACTIVE: This bit indicates the status of the Data Link Control and Management State Machine. It returns a 1b to indicate the DL_Active state, 0b otherwise. This bit must be implemented if the corresponding Data Link Layer Active Capability bit is implemented. Otherwise, this bit must be hardwired to 0b.
28	R	None	T_PCIE2_RP_LINK_CONTROL_STATUS_SLOT_CLOCK_CONFIG: This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference on the connector, this bit must be clear.
27	R	None	T_PCIE2_RP_LINK_CONTROL_STATUS_LINK_TRAINING: This read-only bit indicates that Link training is in progress; hardware clears this bit once training is complete. Note: This field is not applicable and reserved for endpoint devices and Upstream Ports of Switches.

Bit	R/W	Reset	Description
26	R	None	T_PCIE2_RP_LINK_CONTROL_STATUS_UNDEFINED: The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.
25:20	R	None	T_PCIE2_RP_LINK_CONTROL_STATUS_NEG_LINK_WIDTH: This field indicates the negotiated width of the given PCI Express Link. Defined encodings are: 000001b -- x1 000010b -- x2 000100b -- x4 001000b -- x8 001100b -- x12 010000b -- x16 100000b -- x32
19:16	R	1h	T_PCIE2_RP_LINK_CONTROL_STATUS_LINK_SPEED: This field indicates the negotiated Link Speed of the given PCI Express Link. Defined encodings are: 0001b = 2.5 Gb/s PCI Express Link 0010b = 5.0 Gb/s PCI Express Link All other encodings are reserved. 1h: LINK_SPEED_GEN1 (default) 2h: LINK_SPEED_GEN2 (default)
15:12	R	0	Reserved
11	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_AUTO_BANDWIDTH_INT_EN: 0h: AUTO_BANDWIDTH_INT_EN_INIT (default)
10	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_BW_MANAGEMENT_INT_EN: 0h: BW_MANAGEMENT_INT_EN_INIT (default)
9	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_HW_AUTO_WIDTH_DISABLE: 0h: HW_AUTO_WIDTH_DISABLE_DEFAULT (default)
8	R	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_CLOCK_PM: 0h: CLOCK_PM_DEFAULT (default)
7	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_EXTENDED_SYNCH: This bit when set forces extended transmission of FTS ordered sets in FTS and extra TS2 at exit from L1 prior to entering L0. This mode provides external devices monitoring the link time to achieve bit and symbol lock before the link enters L0 state and resumes communication. Default value for this bit is 0. 0h: EXTENDED_SYNCH_INIT (default)
6	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_COMMON_CLOCK_CONFIGURATION: This bit when set indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. A value of 0 indicates that this component and the component at the opposite end of this Link are operating with asynchronous reference clock. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies. 0h: COMMON_CLOCK_CONFIGURATION_INIT (default)
5	R	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_RETRAIN_LINK: This bit initiates Link retraining when set. This field is not applicable and reserved for endpoint devices and Upstream Ports of a Switch. This bit always returns 0 when read. 0h: RETRAIN_LINK_INIT (default)
4	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_LINK_DISABLE: This bit disables the Link when set to 1b. This field is not applicable and reserved for endpoint devices and Upstream Ports of a Switch. Writes to this bit are immediately reflected in the value read from the bit, regardless of the actual Link State. 0h: LINK_DISABLE_INIT (default)
3	R	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_READ_COMPLETION_BOUNDARY: Encodings are: 0b = 64 byte 1b = 128 byte This field is hardwired for a Root Port and returns its RCB support capabilities. Devices that do not implement this feature must hardwire the field to 0b. This field is R/W for devices other than Root Ports. 0h: READ_COMPLETION_BOUNDARY_INIT (default)
2	R	0	Reserved

Bit	R/W	Reset	Description
1:0	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_ACTIVE_STATE_LINK_PM_CONTROL: This field controls the level of active state PM supported on the given PCI Express Link. Defined encodings are: 00b -- Reserved 01b -- L0s Entry Supported 10b -- Reserved 11b -- L0s and L1 Supported 0h: ACTIVE_STATE_LINK_PM_CONTROL_INIT (default)

34.4.5.6 T_PCIE2_RP_SLOT_CAPABILITIES

The Slot Capabilities register identifies PCI Express slot specific capabilities.

Slot Capabilities Register

Offset: 0x94 | Read/Write: RO

Bit	Reset	Description
31:19	None	T_PCIE2_RP_SLOT_CAPABILITIES_PHYSICAL_SLOT_NUMBER: This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18	0h	T_PCIE2_RP_SLOT_CAPABILITIES_NO_CMD_COMPLETED_SUPPORT: When set to 1, this bit indicates that this slot does not generate software notification when an issued command is completed by the Hot Plug Controller. 0h: NO_CMD_COMPLETED_SUPPORT_INIT (default)
17	0h	T_PCIE2_RP_SLOT_CAPABILITIES_ELECTROMECHANICAL_INTERLOCK_PRESENT: Indicates that the ElectroMechanical Interlock mechanism is implemented. 0h: ELECTROMECHANICAL_INTERLOCK_PRESENT_NO (default)
16:15	None	T_PCIE2_RP_SLOT_CAPABILITIES_SLOT_POWER_LIMIT_SCALE: This field Specifies the scale used for the Slot Power Limit Value. Range of values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x This field must be implemented if the Slot Implemented bit is set. The default value prior to hardware/firmware initialization is 00b.
14:7	None	T_PCIE2_RP_SLOT_CAPABILITIES_SLOT_POWER_LIMIT_VALUE: In combination with the Slot Power Limit Scale value, this field specifies the upper limit on power supplied by slot. Power limit (in watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This field must be implemented if the Slot Implemented bit is set. The default value prior to hardware/firmware initialization is 0000_0000b.
6	0h	T_PCIE2_RP_SLOT_CAPABILITIES_HOT_PLUG_CAPABLE: This bit when set indicates that this slot is capable of supporting Hot-plug operations. 0h: HOT_PLUG_CAPABLE_NO (default)
5	0h	T_PCIE2_RP_SLOT_CAPABILITIES_HOT_PLUG_SURPRISE: This bit when set indicates that a device present in this slot might be removed from the system without any prior notification. 0h: HOT_PLUG_SURPRISE_NO (default)
4	0h	T_PCIE2_RP_SLOT_CAPABILITIES_POWER_INDICATOR_PRESENT: This bit when set indicates that a Power Indicator is implemented on the chassis for this slot. 0h: POWER_INDICATOR_PRESENT_NO (default)
3	0h	T_PCIE2_RP_SLOT_CAPABILITIES_ATTENTION_INDICATOR_PRESENT: This bit when set indicates that an Attention Indicator is implemented on the chassis for this slot. 0h: ATTENTION_INDICATOR_PRESENT_NO (default)
2	0h	T_PCIE2_RP_SLOT_CAPABILITIES_MRL_SENSOR_PRESENT: This bit when set indicates that an Attention Indicator is implemented on the chassis for this slot. 0h: MRL_SENSOR_PRESENT_NO (default)
1	0h	T_PCIE2_RP_SLOT_CAPABILITIES_POWER_CONTROLLER_PRESENT: This bit when set indicates that a Power Controller is implemented for this slot. 0h: POWER_CONTROLLER_PRESENT_NO (default)

Bit	Reset	Description
0	0h	T_PCIE2_RP_SLOT_CAPABILITIES_ATTENTION_BUTTON_PRESENT: This bit when set indicates that an Attention Button is implemented on the chassis for this slot. 0h: ATTENTION_BUTTON_PRESENT_NO (default)

34.4.5.7 T_PCIE2_RP_SLOT_CONTROL_STATUS

The Slot Control Register controls PCI Express Slot specific parameters. It also provides information about PCI Express Slot specific parameters.

Slot Control and Slot Status Registers

Offset: 0x98 | Read/Write: R/W

Bit	R/W	Reset	Description
31:25	R	0	Reserved
24	RW1C	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_DL_LAYER_STATE_CHANGED: When set to 1, this field enables software notification when Data Link Layer Active field is changed 0h: DL_LAYER_STATE_CHANGED_INIT (default) 1h: DL_LAYER_STATE_CHANGED_SET
23	R	None	T_PCIE2_RP_SLOT_CONTROL_STATUS_ELECTROMECHANICAL_INTERLOCK_STATE: This bit when set physically locks the add-in card which the user is trying to remove, till software releases it by resetting the bit. Indicates the current state of the interlock, that is, whether the card is electro-mechanically engaged or disengaged 0b: disengaged 1b: engaged Software reads this bit to determine what state the card is in 1h: ELECTROMECHANICAL_INTERLOCK_STATE_YES
22	R	None	T_PCIE2_RP_SLOT_CONTROL_STATUS_PRESENCE_DETECT_STATE: This bit indicates the presence of a card in the slot. The bit reflects the Presence Detect status determined via an in-band mechanism or via the Present Detect pins as defined in the PCI Express Card Electromechanical Specification. Defined encodings are: 0b = Slot Empty. 1b = Card Present in slot. This field is required to be implemented on all Switch devices and Root Ports. The presence detect field for Switch devices or Root Ports not connected to slots should be hardwired to 1. This field is required if a slot is implemented. 1h: PRESENCE_DETECT_STATE_YES
21	R	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_MRL_SENSOR_STATE: This bit reports the status of the MRL sensor if it is implemented. Defined encodings are: 0b = MRL Closed. 1b = MRL Open. 0h: MRL_SENSOR_STATE_INIT (default)
20	RW1C	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_COMMAND_COMPLETED: This bit is set when the hot plug controller completes an issued command. 0h: COMMAND_COMPLETED_INIT (default) 1h: COMMAND_COMPLETED_SET
19	RW1C	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_PRESENCE_DETECT_CHANGED: This bit is set when a Presence Detect change is detected. 0h: PRESENCE_DETECT_CHANGED_INIT (default) 1h: PRESENCE_DETECT_CHANGED_SET
18	RW1C	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_MRL_SENSOR_CHANGED: This bit is set when a MRL Sensor state change is detected. 0h: MRL_SENSOR_CHANGED_INIT (default) 1h: MRL_SENSOR_CHANGED_SET
17	RW1C	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_FAULT_DETECTED: This bit is set when the Power Controller detects a power fault at this slot. 0h: POWER_FAULT_DETECTED_INIT (default) 1h: POWER_FAULT_DETECTED_SET
16	RW1C	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_ATTN_BUTTON_PRESSED: This bit is set when the attention button is pressed. 0h: ATTN_BUTTON_PRESSED_INIT (default) 1h: ATTN_BUTTON_PRESSED_SET
15:13	R	0	Reserved

Bit	R/W	Reset	Description
12	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_DL_LAYER_STATE_CHANGED_ENABLE: This bit is set when the value reported in the Data Link Layer Link Active field of the Link Status register is changed 0h: DL_LAYER_STATE_CHANGED_ENABLE_INIT (default)
11	R	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_ELECTROMECHANICAL_INTERLOCK_CONTROL: indicates that the slot supports electromechanical interlock mechanism 0h: ELECTROMECHANICAL_INTERLOCK_CONTROL_INIT (default)
10	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_CONTROLLER_CONTROL: When read this bit returns the current state of the Power applied to the slot; when written sets the power state of the slot per the defined encodings. 0b = Power On. 1b = Power Off. 0h: POWER_CONTROLLER_CONTROL_INIT (default)
9:8	R/W	1h	T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_INDICATOR_CONTROL: Reads to this bit return the current state of the Power Indicator; writes to this bit set the Power Indicator. 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this bit also cause the Port to send the appropriate POWER_INDICATOR_* messages. 1h: POWER_INDICATOR_CONTROL_INIT (default)
7:6	R/W	3h	T_PCIE2_RP_SLOT_CONTROL_STATUS_ATTN_INDICATOR_CONTROL: Reads to this field return the current state of the Attention Indicator; writes to this field set the Attention Indicator. Defined encodings are: 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this field also cause the Port to send the appropriate ATTENTION_INDICATOR_* messages. 3h: ATTN_INDICATOR_CONTROL_INIT (default)
5	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_HOT_PLUG_INTERRUPT_ENABLE: This bit when set enables generation of hot plug interrupt on enabled hot plug events. 0h: HOT_PLUG_INTERRUPT_ENABLE_INIT (default)
4	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_COMMAND_COMPLETED_INTERRUPT_ENABLE: This bit when set enables the generation of hot plug interrupt when a command is completed by the Hot plug controller. 0h: COMMAND_COMPLETED_INTERRUPT_ENABLE_INIT (default)
3	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_PRESENCE_DETECT_CHANGED_ENABLE: This bit when set enables the generation of hot plug interrupt or wake message on a presence detect changed event. 0h: PRESENCE_DETECT_CHANGED_ENABLE_INIT (default)
2	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_MRL_SENSOR_CHANGED_ENABLE: This bit when set enables the generation of hot plug interrupt or wake message on a MRL sensor changed event. 0h: MRL_SENSOR_CHANGED_ENABLE_INIT (default)
1	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_FAULT_DETECTED_ENABLE: This bit when set enables the generation of hot plug interrupt or wake message on a power fault event. 0h: POWER_FAULT_DETECTED_ENABLE_INIT (default)
0	R/W	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_ATTN_BUTTON_PRESSED_ENABLE: This bit when set enables the generation of hot plug interrupt or wake message on an attention button pressed event. 0h: ATTN_BUTTON_PRESSED_ENABLE_INIT (default)

34.4.5.8 T_PCIE2_RP_RCR

The Root Control Register controls PCI Express Root Complex specific parameters. Bit positions 31:4 of this address are reserved.

Root Control Register

Offset: 0x9c | Read/Write: R/W

Bit	R/W	Reset	Description
31:4	R	0	Reserved
3	R/W	0h	T_PCIE2_RP_RCR_PME_INT: The PME_INT bit is enable bit for generating interrupt upon receipt of a PME message as reflected in the PMESTAT bit of Root Status Register. 0h: PME_INT_DIS (default)
2	R/W	0h	T_PCIE2_RP_RCR_SERR_FAT: The SERR_FAT bit is enable bit for generating System Error if a fatal error (ERR_FATAL) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. 0h: SERR_FAT_DIS (default)
1	R/W	0h	T_PCIE2_RP_RCR_SERR_NONFAT: The SERR_NONFAT bit is enable bit for generating System Error if a non-fatal error (ERR_NONFATAL) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. 0h: SERR_NONFAT_DIS (default)
0	R/W	0h	T_PCIE2_RP_RCR_SERR_COR: The SERR_COR bit is enable bit for generating System Error if a correctable error (ERR_COR) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. 0h: SERR_COR_DIS (default)

34.4.5.9 T_PCIE2_RP_RSR

The Root Status Register provides information about PCI Express device specific parameters. Bit positions 31:18 are reserved.

Root Status Register

Offset: 0xA0 | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R	0	Reserved
17	R	None	T_PCIE2_RP_RSR_PMEPEND: The PMEPEND bit indicates that another PME is pending when the PMESTAT bit is set. When the PMESTAT bit is cleared by software, the PME is delivered by hardware by setting the PMESTAT bit again and updating the REQID appropriately. The PMEPEND bit is cleared by hardware if no more PMEs are pending.
16	RW1C	0h	T_PCIE2_RP_RSR_PMESTAT: The PMESTAT bit indicates that PME was asserted by the requester ID indicated in the PME Requester ID field. Subsequent PMEs are kept pending until the status register is cleared by software by writing a 1. 0h: PMESTAT_NOT_ACTIVE (default) 1h: PMESTAT_ACTIVE 1h: PMESTAT_SET
15:0	R	None	T_PCIE2_RP_RSR_REQID: The REQID field indicates the PCI requester ID of the last PME requester.

34.4.5.10 T_PCIE2_RP_DEVICE_CAPABILITIES_2

Device Capabilities 2 Register

Offset: 0xA4 | Read/Write: RO

Bit	Reset	Description
31:12	0h	T_PCIE2_RP_DEVICE_CAPABILITIES_2_RESERVED_2: Unused bits in this register. 0h: RESERVED_2_INIT (default)
11	1h	T_PCIE2_RP_DEVICE_CAPABILITIES_2_LTR_MECH_SUP: Indicates if LTR Capability is supported. 1h: LTR_MECH_SUP_INIT

Bit	Reset	Description
10:5	0h	T_PCIE2_RP_DEVICE_CAPABILITIES_2_RESERVED: Unused bits in this register. 0h: RESERVED_1_INIT (default)
4	1h	T_PCIE2_RP_DEVICE_CAPABILITIES_2_CPL_TO_DIS_SUP: Support disabling the completion timeout mechanism. 1h: CPL_TO_DIS_SUP_0 (default)
3:0	3h	T_PCIE2_RP_DEVICE_CAPABILITIES_2_CPL_TO_RANGES_SUP: Completion timeout ranges supported by the Root Port. Ranges A and B are currently supported by the PCIe 2.0 design. 3h: CPL_TO_RANGES_SUP_0 (default)

34.4.5.11 T_PCIE2_RP_DEVICE_CONTROL_STATUS_2

Device Control 2 and Device Status 2 Registers

This register is primarily used for Root Port Completion Timeout values. Bits 31:5 are reserved.

Offset: 0xA8 | Read/Write: R/W

Bit	Reset	Description
31:11	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_RESERVED_2: 0h: RESERVED_2_INIT
10	1h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_LTR_MECH_EN: This field is used to enable/disable the LTR Mechanism. By default, the LTR Mechanism is enabled. 1h: LTR_MECH_EN_DEFAULT
9:5	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_RESERVED: Unused bits in this register. 0h: RESERVED_INIT (default)
4	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_CPL_TO_DISABLE: Disables completion timeout, making the Root Port always wait endlessly for a completion to a request. 0h: CPL_TO_DISABLE_DEFAULT (default)
3:0	0h	T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_CPL_TO_VALUE: This field determines the range and hence the timeout values for Completions expected from the endpoint. Range A: 50 µs to 100 ms (80 µs in design) 1 ms to 10 ms (5 ms in design) Range B: 16 ms to 55 ms (40 ms in design) 65 ms to 210 ms (140 ms in design) Default: 50 µs to 50 ms (10 ms in design) Note: When CPL_TO_OVERRIDE is enabled, the value in this field is ignored, and the value in CPL_TO_CUSTOM_VALUE is used. 1h: CPL_TO_VALUE_RANGE_A_LO 2h: CPL_TO_VALUE_RANGE_A_HI 5h: CPL_TO_VALUE_RANGE_B_LO 6h: CPL_TO_VALUE_RANGE_B_HI 0h: CPL_TO_VALUE_DEFAULT (default)

34.4.5.12 T_PCIE2_RP_LINK_CAPABILITIES_2

Link Capabilities 2 Register

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0xac | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_LINK_CAPABILITIES_2_BITS: 0h: BITS_0

34.4.5.13 T_PCIE2_RP_LINK_CONTROL_STATUS_2

Link Control 2 and Link Status 2 Registers

Offset: 0xB0 | Read/Write: R/W

Bit	R/W	Reset	Description
31:17	R	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_RESERVED_STATUS: 0h: RESERVED_STATUS_DEFAULT (default)
16	R	None	T_PCIE2_RP_LINK_CONTROL_STATUS_2_CURRENT_DEEMPHASIS_LEVEL: 1h: CURRENT_DEEMPHASIS_LEVEL_3P5 0h: CURRENT_DEEMPHASIS_LEVEL_6
15:13	R	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_RESERVED_CONTROL: 0h: RESERVED_CONTROL_DEFAULT (default)
12	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_COMPLIANCE_DEEMPHASIS: NOTE: This field is reset by Cold Reset. 0h: COMPLIANCE_DEEMPHASIS_INIT (default)
11	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_COMPLIANCE_SOS: NOTE: This field is reset by Cold Reset. 0h: COMPLIANCE_SOS_INIT (default)
10	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_ENTER_MODIFIED_COMPLIANCE: When this bit is set to 1, the device transmits the modified compliance pattern if the LTSSM enters Polling. Compliance state. Default value is 0b. NOTE: This field is reset by Cold Reset. 0h: ENTER_MODIFIED_COMPLIANCE_INIT (default)
9:7	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_TRANSMIT_MARGIN: This field controls the value of the non-deemphasized voltage level at the transmitter pins. Encodings: 000b: 800-1200mV for full swing and 400-600mV for half-swing 001b-010b: Values must be monotonic with a non-zero slope 011b: 200-400mV for full-swing and 100-200mV for half-swing 100b-111b: reserved Default value is 0b. NOTE: This field is reset by Cold Reset. 0h: TRANSMIT_MARGIN_INIT (default)
6	R	None	T_PCIE2_RP_LINK_CONTROL_STATUS_2_SELECTABLE_DEEMPHASIS: When link is operating at 5GT/s speed, selects the level of de-emphasis. This value is initialized by hardware and/or BIOS. Encodings: 1b: -3.5dB 0b: -6dB Default value is 1b.
5	R	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_HW_AUTO_SPEED_DISABLE: Link speed change is disabled if set to 1b. Default value is 0b. 0h: HW_AUTO_SPEED_DISABLE_INIT
4	R/W	0h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_ENTER_COMPLIANCE: If set to 1b forces link to enter compliance mode at the speed indicated by Target Link Speed field. Default value is 0b. NOTE: This field is reset by Cold Reset. 0h: ENTER_COMPLIANCE_INIT (default)
3:0	R/W	2h	T_PCIE2_RP_LINK_CONTROL_STATUS_2_TARGET_LINK_SPEED: This field sets an upper limit on the link operational speed by restricting the values advertised in training sequences. Defined encodings: 0001b: 2.5GT/s (Gen1) 0010b: 5GT/s (Gen2) Other encodings are reserved. Default value is 0. NOTE: This field is reset by Cold Reset. 0h: TARGET_LINK_SPEED_GEN2_DIS 1h: TARGET_LINK_SPEED_2P5 2h: TARGET_LINK_SPEED_5P0 (default)

34.4.5.14 T_PCIE2_RP_SLOT_CAPABILITIES_2

Slot Capabilities 2 Register

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0xB4 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_SLOT_CAPABILITIES_2_BITS: 0h: BITS_0

34.4.5.15 T_PCIE2_RP_SLOT_CONTROL_STATUS_2

Slot Control 2 and Slot Status 2 Registers

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0xB8 | Read/Write: RO

Bit	Reset	Description
31:0	0h	T_PCIE2_RP_SLOT_CONTROL_STATUS_2_BITS: 0h: BITS_0

34.4.6 Error Reporting Capability Registers

This section lists the PCI Express Advanced Error Reporting Capability List registers as defined in PCI Express Base Specification, Revision 3.0.

34.4.6.1 T_PCIE2_RP_ERPTCAP

PCI Express Extended Capability Header Register

This is the Advanced Error Reporting Enhanced Capability Header register.

Offset: 0x100 | Read/Write: RO

Bit	R/W	Reset	Description
31:20	R	None	T_PCIE2_RP_ERPTCAP_NEXT_PTR: 000h: NONE (default) 140h: L1_SUBSTATE
19:16	R	0h	T_PCIE2_RP_ERPTCAP_VERSION: 0h: VERSION_0 (default) 1h: VERSION_1
15:0	R	0h	T_PCIE2_RP_ERPTCAP_ID: 0h: ID_NONE (default) 1h: ID_AER

34.4.6.2 T_PCIE2_RP_ERPTCAP_UCERR

Uncorrectable Error Status Register

Offset: 0x104 | Read/Write: R/W

Bit	R/W	Reset	Description
31:21	R	0	Reserved
20	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_UNSUP_REQ_ERR: Unsupported Request Error Status NOTE: this field is reset by Cold Reset 0h: UNSUP_REQ_ERR_FALSE (default) 1h: UNSUP_REQ_ERR_TRUE 1h: UNSUP_REQ_ERR_CLEAR

Bit	R/W	Reset	Description
19	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_ECRC_ERR: ECRC Error Status NOTE: this field is reset by Cold Reset 0h: ECRC_ERR_FALSE (default) 1h: ECRC_ERR_TRUE 1h: ECRC_ERR_CLEAR
18	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_MF_TLP: Malformed TLP Status NOTE: this field is reset by Cold Reset 0h: MF_TLP_FALSE (default) 1h: MF_TLP_TRUE 1h: MF_TLP_CLEAR
17	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_RCV_OVFL: Receiver Overflow Status NOTE: this field is reset by Cold Reset 0h: RCV_OVFL_FALSE (default) 1h: RCV_OVFL_TRUE 1h: RCV_OVFL_CLEAR
16	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_UNEXP_COMP: Unexpected Completion Status NOTE: this field is reset by Cold Reset 0h: UNEXP_COMP_FALSE (default) 1h: UNEXP_COMP_TRUE 1h: UNEXP_COMP_CLEAR
15	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_COMP_ABORT: Completion Abort Status NOTE: this field is reset by Cold Reset 0h: COMP_ABORT_FALSE (default) 1h: COMP_ABORT_TRUE 1h: COMP_ABORT_CLEAR
14	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_COMP_TO: Completion Timeout Status NOTE: this field is reset by Cold Reset 0h: COMP_TO_FALSE (default) 1h: COMP_TO_TRUE 1h: COMP_TO_CLEAR
13	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_FC_PROTO_ERR: Flow Control Protocol Error Status NOTE: this field is reset by Cold Reset 0h: FC_PROTO_ERR_DEFAULT (default) 1h: FC_PROTO_ERR_TRUE 1h: FC_PROTO_ERR_CLEAR
12	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_POS_TLP: Poisoned TLP Status NOTE: this field is reset by Cold Reset 0h: POS_TLP_FALSE (default) 1h: POS_TLP_TRUE 1h: POS_TLP_CLEAR
11:5	R	0	Reserved
4	RW1C	0h	T_PCIE2_RP_ERPTCAP_UCERR_DLINK_PROTO_ERR: Data Link Protocol Error Status NOTE: this field is reset by Cold Reset 0h: DLINK_PROTO_ERR_FALSE (default) 1h: DLINK_PROTO_ERR_TRUE 1h: DLINK_PROTO_ERR_CLEAR
3:1	R	0	Reserved
0	R	0h	T_PCIE2_RP_ERPTCAP_UCERR_TRAINING_ERR: Training Error Status 0h: TRAINING_ERR_DEFAULT (default)

34.4.6.3 T_PCIE2_RP_ERPTCAP_UCERR_MK

Uncorrectable Error Mask Register

Offset: 0x108 | Read/Write: R/W

Bit	R/W	Reset	Description
31:21	R	0	Reserved
20	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_UNSUP_REQ_ERR: NOTE: this field is reset by Cold Reset 0h: UNSUP_REQ_ERR_NOT_MASKED (default) 1h: UNSUP_REQ_ERR_MASKED
19	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_ECRC_ERR: NOTE: this field is reset by Cold Reset 0h: ECRC_ERR_NOT_MASKED (default) 1h: ECRC_ERR_MASKED
18	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_MF_TLP: NOTE: this field is reset by Cold Reset 0h: MF_TLP_NOT_MASKED (default) 1h: MF_TLP_MASKED
17	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_RCV_OVFL: NOTE: this field is reset by Cold Reset 0h: RCV_OVFL_NOT_MASKED (default) 1h: RCV_OVFL_MASKED
16	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_UNEXP_COMP: NOTE: this field is reset by Cold Reset 0h: UNEXP_COMP_NOT_MASKED (default) 1h: UNEXP_COMP_MASKED
15	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_COMP_ABORT: NOTE: this field is reset by Cold Reset 0h: COMP_ABORT_NOT_MASKED (default) 1h: COMP_ABORT_MASKED
14	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_COMP_TO: NOTE: this field is reset by Cold Reset 0h: COMP_TO_NOT_MASKED (default) 1h: COMP_TO_MASKED
13	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_FC_PROTO_ERR: NOTE: this field is reset by Cold Reset 0h: FC_PROTO_ERR_NOT_MASKED (default) 1h: FC_PROTO_ERR_MASKED
12	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_POS_TLP: NOTE: this field is reset by Cold Reset 0h: POS_TLP_NOT_MASKED (default) 1h: POS_TLP_MASKED
11:5	R	0	Reserved
4	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_DLINK_PROTO_ERR: NOTE: this field is reset by Cold Reset 0h: DLINK_PROTO_ERR_NOT_MASKED (default) 1h: DLINK_PROTO_ERR_MASKED
3:1	R	0	Reserved
0	R	0h	T_PCIE2_RP_ERPTCAP_UCERR_MK_TRAINING_ERR: 0h: TRAINING_ERR_DEFAULT (default)

34.4.6.4 T_PCIE2_RP_ERPTCAP_UCERR_SEVR

Uncorrectable Error Severity Register

Offset: 0x10C | Read/Write: R/W

Bit	R/W	Reset	Description
31:21	R	0	Reserved

Bit	R/W	Reset	Description
20	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_UNSUP_REQ_ERR: NOTE: this field is reset by Cold Reset 0h: UNSUP_REQ_ERR_NON_FATAL (default) 1h: UNSUP_REQ_ERR_FATAL
19	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_ECRC_ERR: NOTE: this field is reset by Cold Reset 0h: ECRC_ERR_NON_FATAL (default) 1h: ECRC_ERR_FATAL
18	R/W	1h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_MF_TLP: NOTE: this field is reset by Cold Reset 0h: MF_TLP_NON_FATAL 1h: MF_TLP_FATAL (default)
17	R/W	1h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_RCV_OVFL: NOTE: this field is reset by Cold Reset 0h: RCV_OVFL_NON_FATAL 1h: RCV_OVFL_FATAL (default)
16	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_UNEXP_COMP: NOTE: this field is reset by Cold Reset 0h: UNEXP_COMP_NON_FATAL (default) 1h: UNEXP_COMP_FATAL
15	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_COMP_ABORT: NOTE: this field is reset by Cold Reset 0h: COMP_ABORT_NON_FATAL (default) 1h: COMP_ABORT_FATAL
14	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_COMP_TO: NOTE: this field is reset by Cold Reset 0h: COMP_TO_NON_FATAL (default) 1h: COMP_TO_FATAL
13	R/W	1h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_FC_PROTO_ERR: NOTE: this field is reset by Cold Reset 0h: FC_PROTO_ERR_NON_FATAL 1h: FC_PROTO_ERR_FATAL (default)
12	R/W	0h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_POS_TLP: NOTE: this field is reset by Cold Reset 0h: POS_TLP_NON_FATAL (default) 1h: POS_TLP_FATAL
11:5	R	0	Reserved
4	R/W	1h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_DLINK_PROTO_ERR: NOTE: this field is reset by Cold Reset 0h: DLINK_PROTO_ERR_NON_FATAL 1h: DLINK_PROTO_ERR_FATAL (default)
3:1	R	0	Reserved
0	R	1h	T_PCIE2_RP_ERPTCAP_UCERR_SEVR_TRAINING_ERR: 0h: TRAINING_ERR_NON_FATAL 1h: TRAINING_ERR_FATAL

34.4.6.5 T_PCIE2_RP_ERPTCAP_CERR

Correctable Error Status Register

Offset: 0x110 | Read/Write: R/W

Bit	R/W	Reset	Description
31:14	R	0	Reserved
13	RW1C	0h	T_PCIE2_RP_ERPTCAP_CERR_ADVISORY_NF: 0h: ADVISORY_NF_NOT_MASKED 1h: ADVISORY_NF_MASKED (default)
12	RW1C	0h	T_PCIE2_RP_ERPTCAP_CERR_RPLY_TO: 0h: RPLY_TO_NOT_MASKED (default) 1h: RPLY_TO_MASKED
11:9	R	0	Reserved

Bit	R/W	Reset	Description
8	RW1C	0h	T_PCIE2_RP_ERPTCAP_CERR_RPLY_RLOV: 0h: RPLY_RLOV_NOT_MASKED (default) 1h: RPLY_RLOV_MASKED
7	RW1C	0h	T_PCIE2_RP_ERPTCAP_CERR_BAD_DLLP: 0h: BAD_DLLP_NOT_MASKED (default) 1h: BAD_DLLP_MASKED
6	RW1C	0h	T_PCIE2_RP_ERPTCAP_CERR_BAD_TLP: 0h: BAD_TLP_NOT_MASKED (default) 1h: BAD_TLP_MASKED
5:1	R	0	Reserved
0	RW1C	0h	T_PCIE2_RP_ERPTCAP_CERR_RCV_ERR: 0h: RCV_ERR_NOT_MASKED (default) 1h: RCV_ERR_MASKED

34.4.6.6 T_PCIE2_RP_ERPTCAP_CERR_MK

Correctable Error Mask Register

Offset: 0x114 | Read/Write: R/W

Bit	R/W	Reset	Description
31:14	R	0	Reserved
13	R/W	1h	T_PCIE2_RP_ERPTCAP_CERR_MK_ADVISORY_NF: 0h: ADVISORY_NF_NOT_MASKED 1h: ADVISORY_NF_MASKED (default)
12	R/W	0h	T_PCIE2_RP_ERPTCAP_CERR_MK_RPLY_TO: 0h: RPLY_TO_NOT_MASKED (default) 1h: RPLY_TO_MASKED
11:9	R	0	Reserved
8	R/W	0h	T_PCIE2_RP_ERPTCAP_CERR_MK_RPLY_RLOV: 0h: RPLY_RLOV_NOT_MASKED (default) 1h: RPLY_RLOV_MASKED
7	R/W	0h	T_PCIE2_RP_ERPTCAP_CERR_MK_BAD_DLLP: 0h: BAD_DLLP_NOT_MASKED (default) 1h: BAD_DLLP_MASKED
6	R/W	0h	T_PCIE2_RP_ERPTCAP_CERR_MK_BAD_TLP: 0h: BAD_TLP_NOT_MASKED (default) 1h: BAD_TLP_MASKED
5:1	R	0	Reserved
0	R/W	0h	T_PCIE2_RP_ERPTCAP_CERR_MK_RCV_ERR: 0h: RCV_ERR_NOT_MASKED (default) 1h: RCV_ERR_MASKED

34.4.6.7 T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL

Advanced Error Capabilities and Control Register

Offset: 0x118 | Read/Write: R

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	0h	T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_CHK_EN: NOTE: this field is reset by Cold Reset 0h: ECRC_CHK_EN_FALSE (default) 1h: ECRC_CHK_EN_TRUE
7	R	1h	T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_CHK_CAP: 1h: ECRC_CHK_CAP_TRUE (default)
6	R/W	0h	T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_GEN_EN: NOTE: this field is reset by Cold Reset 0h: ECRC_GEN_EN_FALSE (default) 1h: ECRC_GEN_EN_TRUE

Bit	R/W	Reset	Description
5	R	1h	T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_GEN_CAP: 1h: ECRC_GEN_CAP_TRUE (default)
4:0	R	None	T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ERR_PTR:

34.4.6.8 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0

This register is 16 bytes wide. The header is captured such that the fields of the header read by software in the same way the headers are presented in the specification, when the register is read using Dword accesses. Therefore, byte 0 of the header is located in byte 3 of the Header Log register, byte 1 of the header is in the byte 2 of the header Log register, and so forth.

Header Log Register

Offset: 0x11C | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_0:

34.4.6.9 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW1

Offset: 0x120 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW1_1:

34.4.6.10 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW2

Offset: 0x124 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW2_2:

34.4.6.11 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW3

Offset: 0x128 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	T_PCIE2_RP_ERPTCAP_HDR_LOG_DW3_3:

34.4.6.12 T_PCIE2_RP_ERPTCAP_ERR_CMD

Root Error Command Register

Offset: 0x12C | Read/Write: R/W

Bit	R/W	Reset	Description
31:3	R	0	Reserved
2	R/W	0h	T_PCIE2_RP_ERPTCAP_ERR_CMD_FATAL_ERR_RPT_EN: 0h: FATAL_ERR_RPT_EN_FALSE (default) 1h: FATAL_ERR_RPT_EN_TRUE
1	R/W	0h	T_PCIE2_RP_ERPTCAP_ERR_CMD_NONFATAL_ERR_RPT_EN: 0h: NONFATAL_ERR_RPT_EN_FALSE (default) 1h: NONFATAL_ERR_RPT_EN_TRUE
0	R/W	0h	T_PCIE2_RP_ERPTCAP_ERR_CMD_COR_ERR_RPT_EN: 0h: COR_ERR_RPT_EN_FALSE (default) 1h: COR_ERR_RPT_EN_TRUE

34.4.6.13 T_PCIE2_RP_ERPTCAP_ERR_STS

Root Error Status Register

Offset: 0x130 | Read/Write: R/W

Bit	R/W	Reset	Description
31:27	R	None	T_PCIE2_RP_ERPTCAP_ERR_STS_ADV_ERR_INTR_MSG_NUM: If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base message data and the MSI message that is generated when any of the status bits of this capability are set. Hardware is required to update this field so that it is correct if the number of MSI messages assigned to the device changes. 1h: ADV_ERR_INTR_MSG_NUM_ONE
26:7	R/W	0	Reserved
6	RW1C	0h	T_PCIE2_RP_ERPTCAP_ERR_STS_FATAL_RCVD: Set when one or more fatal uncorrectable error messages have been received. NOTE: this field is reset by Cold Reset 0h: FATAL_RCVD_FALSE (default) 1h: FATAL_RCVD_TRUE 1h: FATAL_RCVD_CLEAR
5	RW1C	0h	T_PCIE2_RP_ERPTCAP_ERR_STS_NONFATAL_RCVD: Set when one or more non-fatal uncorrectable error messages have been received. NOTE: this field is reset by Cold Reset 0h: NONFATAL_RCVD_FALSE (default) 1h: NONFATAL_RCVD_TRUE 1h: NONFATAL_RCVD_CLEAR
4	RW1C	0h	T_PCIE2_RP_ERPTCAP_ERR_STS_FIRST_FATAL_RCVD: Set when the first uncorrectable error message received is for a fatal error. NOTE: this field is reset by Cold Reset 0h: FIRST_FATAL_RCVD_FALSE (default) 1h: FIRST_FATAL_RCVD_TRUE 1h: FIRST_FATAL_RCVD_CLEAR
3	RW1C	0h	T_PCIE2_RP_ERPTCAP_ERR_STS_MULT_UNCOR_RCVD: Set when either a fatal or a non-fatal error is received and UNCOR_RCVD is already set. NOTE: this field is reset by Cold Reset 0h: MULT_UNCOR_RCVD_FALSE (default) 1h: MULT_UNCOR_RCVD_TRUE 1h: MULT_UNCOR_RCVD_CLEAR
2	RW1C	0h	T_PCIE2_RP_ERPTCAP_ERR_STS_UNCOR_RCVD: Set when either a fatal or a non-fatal error message is received and this bit is not already set. NOTE: this field is reset by Cold Reset 0h: UNCOR_RCVD_FALSE (default) 1h: UNCOR_RCVD_TRUE 1h: UNCOR_RCVD_CLEAR
1	RW1C	0h	T_PCIE2_RP_ERPTCAP_ERR_STS_MULT_COR_RCVD: Set when a correctable error message is received and COR_RCVD is already set. NOTE: this field is reset by Cold Reset 0h: MULT_COR_RCVD_FALSE (default) 1h: MULT_COR_RCVD_TRUE 1h: MULT_COR_RCVD_CLEAR
0	RW1C	0h	T_PCIE2_RP_ERPTCAP_ERR_STS_COR_RCVD: Set when a correctable error message is received and this bit is not already set. NOTE: this field is reset by Cold Reset 0h: COR_RCVD_FALSE (default) 1h: COR_RCVD_TRUE 1h: COR_RCVD_CLEAR

34.4.6.14 T_PCIE2_RP_ERPTCAP_ERR_ID

This register identifies the source (Requester ID) of first correctable and uncorrectable (non-fatal/fatal) errors reported in the Root Error Status register.

Root Error Status Register

Offset: 0x134 | Read/Write: R

Bit	R/W	Reset	Description
31:16	R	None	T_PCIE2_RP_ERPTCAP_ERR_ID_ERR_COR: 0h: ERR_COR_DEFAULT (default)
15:0	R	None	T_PCIE2_RP_ERPTCAP_ERR_ID_ERR_UNCOR: 0h: ERR_UNCOR_DEFAULT (default)

34.4.7 L1 PM Substate Capability Registers

This section lists the PCI Express L1 PM Substate Capability List registers as defined in L1 PM Substate ECN.

34.4.7.1 T_PCIE2_L1_PM_SUBSTATES_EXT_CAP

PCI Express Extended Capability Header Register

Offset: 0x140 | Read/Write: R

Bit	R/W	Reset	Description
31:20	R	0h	T_PCIE2_RP_L1_PM_SUBSTATES_EXT_CAP_NEXT_CAP: This field contains the offset to the next PCI Express Capability structure. 0h: NULL
19:16	R	1h	T_PCIE2_RP_L1_PM_SUBSTATES_EXT_CAP_VERSION: This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. 1h: L1_SUBSTATES
15:0	R	1Eh	T_PCIE2_RP_L1_PM_SUBSTATES_EXT_CAP: This field is a PCISIG defined ID number that indicates the nature and format of the Extended Capability. 1Eh: ID_L1_SUBSTATES

34.4.7.2 T_PCIE2_L1_PM_SUBSTATES_CAP

L1 PM Substates Capability Register

Offset: 0x144 | Read/Write: R

Bit	R/W	Reset	Description
23:19	R	00101b	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_T_PWRON_VALUE: This register along with the Port T_POWER_ON Scale field in the L1 PM Substates Capabilities register sets the time (in µs) that this Port requires the port on the opposite side of Link to wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface. The value of Port T_POWER_ON is calculated by multiplying the value in this field by the scale value in the Port T_POWER_ON Scale field in the L1 PM Substates Capabilities register. Default value is 00101b Required for all Ports that support L1.2, otherwise this field is of type RsvdP. This register is driven by T_PCIE2_RP_L1_PM_SUBSTATES_CYA_T_PWRON_VALUE if L1.2 is supported else this register is driven with 0's.
17:16	R	00b	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_T_PWRON_SCALE: Specifies the scale used for the Port T_POWER_ON Value field in the L1 PM Substates Capabilities register. 00b = 2 µs (default) 01b = 10 µs 10b = 100 µs 11b = Reserved Required for all Ports that support L1.2, otherwise this field is of type RsvdP. This field is driven by T_PCIE2_RP_L1_PM_SUBSTATES_CYA_T_PWRON_SCALE if L1.2 is supported else this field is driven with 0's.

Bit	R/W	Reset	Description
15:8	R	0xFF	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_CM_RESTORE_TIME: Time (in µs) required for this Port to re-establish common mode. Required for all Ports that support L1.2, otherwise this field is of type RsvdP. This field is driven by T_PCIE2_RP_L1_PM_SUBSTATES_CYA_CM_RESTORE_TIME if L1.2 is supported else this field is driven with 0's. The value ranges from 0 to 255 µs (the units are in microseconds), default setting is 0xFF(255)
4	R	1h	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_L1_PM_SUBSTATES_SUPPORTED: When set this bit indicates that this Port supports L1 PM Substates. For compatibility with possible future extensions, software must not enable L1 PM Substates unless this bit is set. 0h: DISABLE 1h: ENABLE
3	R	1h	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_ASPM_L1_1_SUPPORTED: When set this bit indicates that ASPM L1.1 is supported. 0h: DISABLE 1h: ENABLE
2	R	1h	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_ASPM_L1_2_SUPPORTED: When set this bit indicates that ASPM L1.2 is supported. 0h: DISABLE 1h: ENABLE
1	R	1h	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_PCI_PM_L1_1_SUPPORTED: When set this bit indicates that PCI-PM L1.1 is supported. 0h: DISABLE 1h: ENABLE
0	R	1h	T_PCIE2_RP_L1_PM_SUBSTATES_CAP_PCI_PM_L1_2_SUPPORTED: When set this bit indicates that PCI-PM L1.2 is supported. 0h: DISABLE 1h: ENABLE

34.4.7.3 T_PCIE2_RP_L1_PM_SUBSTATES_CTRL1

L1 PM Substates Control 1 Register

Offset: 0x148 | Read/Write: R/W

Bit	R/W	Reset	Description
31:29	R/W	000b	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL1_LTR_L1_2_THRES_SCALE: This field provides a scale for the value contained within the LTR_L1.2_THRESHOLD_Value. Encoding is the same as the LatencyScale fields in the LTR Message. Hardware operation is undefined if software writes a Not-Permitted value to this field. Required for all Ports that support ASPM L1.2; otherwise this field is of type RsvdP. 000b: INIT
25:16	R/W	0000000000b	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL1_LTR_L1_2_THRES_VAL: Along with the LTR_L1.2_THRESHOLD_Scale, this field indicates the LTR threshold used to determine if entry into L1 results in L1.1 (if enabled) or L1.2 (if enabled). Required for all Ports that support ASPM L1.2; otherwise this field is of type RsvdP. 0000000000b: INIT
15:8	R/W	FFh	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL1_CM_RESTORE_TIME: FFh: INIT
3	R/W	0b	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL1_ASPM_L1_1_EN: When set this bit enables ASPM L1.1. Required for both Upstream and Downstream Ports. For Ports that do not support ASPM L1.1, this bit is permitted to be hardwired to 0. 0b: INIT
2	R/W	0b	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL1_ASPM_L1_2_EN: When set this bit enables ASPM L1.2. Required for both Upstream and Downstream Ports. For Ports that do not support ASPM L1.2, this bit is permitted to be hardwired to 0. 0b: INIT
1	R/W	0b	T_ PCIE2_RP_L1_PM_SUBSTATES_CTRL1_PCIPM_L1_1_EN: When set this bit enables PCI-PM L1.1. Required for both Upstream and Downstream Ports. 0b: INIT

Bit	R/W	Reset	Description
0	R/W	0b	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL1_PCIPM_L1_2_EN: When set this bit enables PCI-PM L1.2. Required for both Upstream and Downstream Ports. For Ports that do not support PCI-PM L1.2 this bit is permitted to be hardwired to 0. 0b: INIT

34.4.7.4 T_PCIE2_RP_L1_PM_SUBSTATES_CTRL2

L1 PM Substates Control 2 Register

Offset: 0x14c | Read/Write: R/W

Bit	R/W	Reset	Description
7:3	R/W	00101b	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL2_T_PWRON_VALUE: Along with the T_POWER_ON Scale sets the minimum amount of time (in us) that the Port must wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface. T_POWER_ON is calculated by multiplying the value in this field by the value in the T_POWER_ON Scale field. Required for all Ports that support L1.2; otherwise this field is of type RsvdP. 00101b: INIT
1:0	R/W	00b	T_PCIE2_RP_L1_PM_SUBSTATES_CTRL2_T_PWRON_SCALE: Specifies the scale used for the Port T_POWER_ON Value field in the L1 PM Substates Capabilities register. 00b = 2 µs 01b = 10 µs 10b = 100 µs 11b = Reserved Required for all Ports that support L1.2; otherwise this field is of type RsvdP. 00b: INIT

34.4.8 NVIDIA Private Registers

The following sections list the private registers defined to configure the operation of NVIDIA® PCIE Root Port controller. These registers are defined outside of the standard PCI Capabilities and PCI Express Extended Capabilities lists. Thus, they cannot be discovered and traversed by standard PCI/PCIE drivers.

34.4.8.1 T_PCIE2_RP_LTR_REP_VAL

L1 PM Substates Diagnostics Register

This register captures the latest LTR value reported by the endpoint. LTR messages are decoded and the LTR values for no snoop and snoop reported in the message are stored in this register. Values are valid only if LTR capability is enabled.

Offset: 0xC10 | Read/Write: R

Bit	R/W	Reset	Description
31:16	R	None	T_PCIE2_RP_LTR_REP_VAL_NO_SNOOP: 0h: NO_SNOOP_INIT
15:0	R	None	T_PCIE2_RP_LTR_REP_VAL_SNOOP: 0h: SNOOP_INIT

34.4.8.2 T_PCIE2_RP_L1_1_ENTRY_COUNT

This register reflects the number of times the link entered the L1.1.

Offset: 0xC14 | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0	T_PCIE2_RP_L1_1_ENTRY_COUNT_RESET: When set to 1 by software, it will clear the T_PCIE2_RP_L1_1_ENTRY_COUNT_VALUE. Will be set to 0 by hardware once the T_PCIE2_RP_L1_1_ENTRY_COUNT_VALUE is cleared. 0: RESET_INIT 0: RESET_DONE (default) 1: RESET_PENDING
30	R/W	0	T_PCIE2_RP_L1_1_ENTRY_COUNT_FREEZE: 0: FREEZE_INIT (default)
29:16	R	0	Reserved
15:0	R	0	T_PCIE2_RP_L1_1_ENTRY_COUNT_VALUE: This field reflects the number of times the link entered the L1.1 state. 0h: VALUE_INIT (default)

34.4.8.3 T_PCIE2_RP_L1_2_ENTRY_COUNT

This register reflects the number of times the link entered the L1.2_ENTRY state, but was forced to exit to L1.0 and then L0 because it saw clkreq# asserted while it was in this state.

Offset: 0xC18 | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0	T_PCIE2_RP_L1_2_ENTRY_COUNT_RESET: When set to 1 by software, it will clear the T_PCIE2_RP_L1_2_ENTRY_COUNT_VALUE. Will be set to 0 by hardware once the T_PCIE2_RP_L1_2_ENTRY_COUNT_VALUE is cleared. 0: RESET_INIT 0: RESET_DONE (default) 1: RESET_PENDING
30	R/W	0	T_PCIE2_RP_L1_2_ENTRY_COUNT_FREEZE: 0: FREEZE_INIT (default)
29:16	R	0	Reserved
15:0	R	0	T_PCIE2_RP_L1_2_ENTRY_COUNT_VALUE: This field reflects the number of times the link entered the L1.2 ENTRY state, but was forced to exit to L1.0 and then L0 because it saw clkreq# asserted while it was in this state. 0h: VALUE_INIT (default)

34.4.8.4 T_PCIE2_RP_L1_2_ABORT_COUNT

Offset: 0xC1C | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0	T_PCIE2_RP_L1_2_ABORT_COUNT_RESET: When set to 1 by software, it will clear the T_PCIE2_RP_L1_2_ABORT_COUNT_VALUE. Will be set to 0 by hardware once the T_PCIE2_RP_L1_2_ABORT_COUNT_VALUE is cleared. 0: RESET_INIT 0: RESET_DONE (default) 1: RESET_PENDING
30	R/W	0	T_PCIE2_RP_L1_2_ABORT_COUNT_FREEZE: 0: FREEZE_INIT (default)
29:16	R	0	Reserved
15:0	R	0	T_PCIE2_RP_L1_2_ABORT_COUNT_VALUE: This field reflects the number of times the link entered the L1.2 ENTRY state, but was forced to exit to L1.0 and then L0 because it saw clkreq# asserted while it was in this state. 0h: VALUE_INIT (default)

34.4.8.5 T_PCIE2_RP_PIPE_CTL

This register controls (enabling /disabling) the pipelines added to meet timing on various interfaces. A value of 0 disables the extra pipeline.

Offset: 0xD5C | Read/Write: R/W

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	1h	T_PCIE2_RP_PIPE_CTL_UFA2WRR_PWTOP: 1h: UFA2WRR_PWTOP_INIT (default)
7	R/W	1h	T_PCIE2_RP_PIPE_CTL_UBFI2DFI_P2P: 1h: UBFI2DFI_P2P_INIT (default)
6	R/W	1h	T_PCIE2_RP_PIPE_CTL_TXBA2DFI_WR: 1h: TXBA2DFI_WR_INIT (default)
5	R/W	1h	T_PCIE2_RP_PIPE_CTL_CMDQ2UFARB: 1h: CMDQ2UFARB_INIT (default)
4	R/W	1h	T_PCIE2_RP_PIPE_CTL_UBFI2DFI_NTT: 1h: UBFI2DFI_NTT_INIT (default)
3	R/W	1h	T_PCIE2_RP_PIPE_CTL_PCA: 1h: PCA_INIT (default)
2	R/W	1h	T_PCIE2_RP_PIPE_CTL_DFIREQ: 1h: DFIREQ_INIT (default)
1	R/W	1h	T_PCIE2_RP_PIPE_CTL_DFIRSP: 1h: DFIRSP_INIT (default)
0	R/W	1h	T_PCIE2_RP_PIPE_CTL_DFI2UBFI: 1h: DFI2UBFI_INIT (default)

34.4.8.6 T_PCIE2_RP_RX_HDR_LIMIT

These register defines the upstream header logical partition sizes. These buffers reside in UBFI. RXL provides the write interface. The buffers constitute asynchronous boundary of xclk -> ufpci_clk.

Offset: 0xE00 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	T_PCIE2_RP_RX_HDR_LIMIT_CPL: Programs the size of the Completion Header Buffer. This field must be programmed with an even value. 0h: CPL_INIT (default)
15:8	R/W	0h	T_PCIE2_RP_RX_HDR_LIMIT_PW: Programs the size of the Posted Write Header Buffer. When ISO PWs are enabled, this field must be halved, and programmed before both ufpci_reset_ and xreset_. This field must be programmed with an even value. 0h: PW_INIT (default)
7:0	R/W	0h	T_PCIE2_RP_RX_HDR_LIMIT_NP: Programs the size of the Non-Posted (Read and Write) Header Buffer. When ISO NPs are enabled, this field must be halved, and programmed before both ufpci_reset_ and xreset_. This field must be programmed with an even value. 0h: NP_INIT (default)

34.4.8.7 T_PCIE2_RP_RX_DATA_LIMIT

These buffers reside in UBFI. RXL provides the write interface. The buffers constitute asynchronous boundary of xclk -> ufpci_clk.

Offset: 0xE04 | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:20	R/W	0h	T_PCIE2_RP_RX_DATA_LIMIT_CPL: Programs the size of the Completions Data Buffer. This field must be programmed with an even value. 0h: CPL_INIT (default)

Bit	R/W	Reset	Description
19:8	R/W	0h	T_PCIE2_RP_RX_DATA_LIMIT_PW: Programs the size of the Posted Writes Data Buffer. When ISO PWs are enabled, this field must be halved, and programmed before both ufpci_reset_ and xreset_. This field must be programmed with an even value. 0h: PW_INIT (default)
7:0	R/W	0h	T_PCIE2_RP_RX_DATA_LIMIT_NP: Programs the size of the Non-Posted Write Header Buffer. This field must be programmed with an even value. 0h: NP_INIT (default)

34.4.8.8 T_PCIE2_RP_TX_HDR_LIMIT

These buffers reside in TXBA. DFI provides the write interface. These buffers constitute an asynchronous boundary of dfpci_clk -> xclk.

Offset: 0xE08 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	0	T_PCIE2_RP_TX_HDR_LIMIT_NPT: 0h: NPT_INIT (default)
23:16	R/W	0h	T_PCIE2_RP_TX_HDR_LIMIT_CPL: Programs the size of the downstream Completions Header Buffer. This must be programmed with a value greater than 3 (or left alone at its default value of 0x0). 0h CPL_INIT (default)
15:8	R/W	0h	T_PCIE2_RP_TX_HDR_LIMIT_PW: Programs the size of the downstream Posted Writes Header Buffer. 0h: PW_INIT (default)
7:0	R/W	0h	T_PCIE2_RP_TX_HDR_LIMIT_NP: Programs the size of the downstream Non Posted Header Buffer. This should be programmed with the same value as TX_DATA_LIMIT_NP. 0h: NP_INIT (default)

34.4.8.9 T_PCIE2_RP_TX_DATA_LIMIT

These buffers reside in TXBA. DFI provides the write interface. These buffers constitute an asynchronous boundary of dfpci_clk -> xclk.

Offset: 0xE0C | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:8	R/W	0h	T_PCIE2_RP_TX_DATA_LIMIT_PW: Programs the size of the downstream Posted Writes Data Buffer. 0h: PW_INIT (default)
7:0	R/W	0h	T_PCIE2_RP_TX_DATA_LIMIT_NP: Programs the size of the downstream Non Posted Data Buffer. This should be programmed with same value as TX_HDR_LIMIT_NP. 0h: NP_INIT (default)

34.4.8.10 T_PCIE2_RP_UFPCI

This register controls upstream FPCI control and arbitration in UBFI and performance fields.

Offset: 0xE10 | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:23	R/W	0h	T_PCIE2_RP_UFPCI_ISO_WEIGHT: 0h: ISO_WEIGHT_INIT (default)
22	R/W	0h	T_PCIE2_RP_UFPCI_ISO_CONTROL_ENABLE: 0h: ISO_CONTROL_ENABLE_INIT (default)

Bit	R/W	Reset	Description
21	R/W	0h	T_PCIE2_RP_UFPCI_ISOPW2HPISO: If set, converts posted writes with TC >= tc2isomap to HP_ISO writes instead of ISO writes. 0h: ISOPW2HPISO_INIT (default)
20	R/W	0h	T_PCIE2_RP_UFPCI_ISONP2HPISO: If set, converts nonposted reads with TC >= tc2isomap to HP_ISO reads instead of ISO reads. 0h: ISONP2HPISO_INIT (default)
19:12	R/W	0h	T_PCIE2_RP_UFPCI_REQ_PEND_PERIOD: The following timers are reset when they hit the value in REQ_PEND_PERIOD 1. noniso_timer - feeds into tms*2sm_iso_pending if system_stutter is supported 2. iso_timer - feeds into tms*2sm_noniso_pending if system_stutter is supported 3. coh_timer - feeds into tms*2sm_coh_request_pend 4. noncoh_timer - feeds into tms*2sm_noncoh_request_pend 0h: REQ_PEND_PERIOD_INIT (default)
11:10	R/W	1h	T_PCIE2_RP_UFPCI_WRR_GRANT_BURST: This field controls the length of time for which a particular controller gets arbitration within a TMS. It relates to time multiplexed inter-TMS arbitration for controllers within a TMS. 1h: WRR_GRANT_BURST_INIT (default)
9:5	R/W	0h	T_PCIE2_RP_UFPCI_PW_PRI_OVR_COUNT: If a posted write has received this number of grants when the priority was swapped to posted write, its priority will be reset. 0h: PW_PRI_OVR_COUNT_INIT (default)
4:0	R/W	0h	T_PCIE2_RP_UFPCI_PW_STARV_COUNT: If an NP has received this number of grants over a posted write while there are posted writes pending, then swap the priority to the posted write. 0h: PW_STARV_COUNT_INIT (default)

34.4.8.11 T_PCIE2_RP_MISC0

This register controls miscellaneous fields.

Offset: 0xE14 | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	T_PCIE2_RP_MISC0_SHORT_RXL_TIMER: This bit is used for simulations only, to test the completion timeout feature in RXL. It shortens the wait period for a completion to 1 µs. 0h: SHORT_RXL_TIMER_INIT (default)
30	R/W	0h	T_PCIE2_RP_MISC0_P2P_SMALL_ISA_HOLE: When set, this bit controls the peer2peer settings on address ranges 64'hA_0000 to 64'hF_FFFF. It is used to omit address ranges A, B, C, D, E and F from being peer2peer. 0h: P2P_SMALL_ISA_HOLE_INIT (default)
29	R/W	0h	T_PCIE2_RP_MISC0_P2P_F: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hF_0000 and 64'hF_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h: P2P_F_INIT (default)
28	R/W	0h	T_PCIE2_RP_MISC0_P2P_E: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hE_0000 and 64'hE_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h: P2P_E_INIT (default)
27	R/W	0h	T_PCIE2_RP_MISC0_P2P_D: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hD_0000 and 64'hD_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h: P2P_D_INIT (default)
26	R/W	0h	T_PCIE2_RP_MISC0_P2P_C: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hC_0000 and 64'hC_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h: P2P_C_INIT (default)
25	R/W	0h	T_PCIE2_RP_MISC0_P2P_B: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hB_0000 and 64'hB_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h: P2P_B_INIT (default)

Bit	R/W	Reset	Description
24	R/W	0h	T_PCIE2_RP_MISC0_P2P_A: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hA_0000 and 64'hA_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h: P2P_A_INIT (default)
23	R/W	0h	T_PCIE2_RP_MISC0_NISONC2HPISO: When set, all upstream Noniso, Non-coherent, Non-peer2peer reads will be upgraded to HPISO Reads. This upgrade is available for all controllers in a TMS if any one of the controllers has iso buffers. 0h: NISONC2HPISO_INIT (default)
22	R	0	Reserved
21	R/W	1h	T_PCIE2_RP_MISC0_AUTO_XCLK_FREQ_EN: When set, the xclk for a TMS will switch dynamically (by hardware itself) between 250 and 500. 250: if all root ports in that TMS are in Gen1. 500: If any root port in that TMS is in Gen2. This bit is valid only for Root Port 0 of that TMS, and don't care for others. 1h: AUTO_XCLK_FREQ_EN_INIT (default)
20	R/W	0h	T_PCIE2_RP_MISC0_RXL_CLEAR_DROP: Clears the DROP ALL status of the receiver. DROP ALL occurs on fatal errors such as Receiver Overflow and Malformed TLPs. Once set, RXL will not allow any other transactions upstream. 0h: RXL_CLEAR_DROP_INIT (default)
19:4	R/W	FFh	T_PCIE2_RP_MISC0_P2P_BURST_SIZE: FFh: P2P_BURST_SIZE_INIT (default)
3	R/W	0h	T_PCIE2_RP_MISC0_ISO_PW_ENABLE: Enables ISO posted write transactions for packets with TC >= TC2ISOMAP, on controllers that have iso buffers. 0h: ISO_PW_ENABLE_INIT (default)
2	R/W	0h	T_PCIE2_RP_MISC0_ISO_NP_ENABLE: Enables ISO NP transactions for packets with TC >= TC2ISOMAP, on controllers that have iso buffers. 0h: ISO_NP_ENABLE_INIT (default)
1	R/W	0h	T_PCIE2_RP_MISC0_NATIVE_P2P_ENABLE: Enables native peer2peer transactions on controllers that support it. Posted Write peer2peer transactions will use a shortcut through the design to enable high performance peer2peer accesses. 0h: NATIVE_P2P_ENABLE_INIT (default)
0	R/W	0h	T_PCIE2_RP_MISC0_ENABLE_CLUMPING: When set, enables unitid clumping on controllers that support it. Controllers that support unitid clumping will resort to using its unitid+1 and an additional 32 source tags when it has exhausted its allotted use of first 32 source tags. 0h: ENABLE_CLUMPING_INIT (default)

34.4.8.12 T_PCIE2_RP_TXBA0

This register is specific to the TXBA sub-unit in PCIe 2.0 and controls:

- Replay buffer limits
- Completion merging limits
- Replay timer control fields

Offset: 0xE18 | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	0h	T_PCIE2_RP_TXBA0_REPLAY_TIMER_EXPIRY: When TXBA0_USE_REPLAY_TIMER_OFFSET is 1, the value programmed into TXBA0_REPLAY_TIMER_EXPIRY is added to the replay timer that hardware calculates. When this bit is 0, the value in TXBA0_REPLAY_TIMER_EXPIRY is selected (if its non-zero) instead of the hardware value. 0h: REPLAY_TIMER_EXPIRY_INIT (default)
15:13	R	0	Reserved
12	R/W	0h	T_PCIE2_RP_TXBA0_USE_REPLAY_TIMER_OFFSET: 0h: USE_REPLAY_TIMER_OFFSET_INIT (default)

Bit	R/W	Reset	Description
11	R/W	0h	T_PCIE2_RP_TXBA0_CMPL_MERGE_UPTO_64DW: Similar to above field, but the upper limit is 64 DW. When neither 32DW/64DW is set, nor completion merging is enabled, maximum merging supported is 128DW. 0h: CMPL_MERGE_UPTO_64DW_INIT (default)
10	R/W	0h	T_PCIE2_RP_TXBA0_CMPL_MERGE_UPTO_32DW: When set, split completions that have payloads up to 32 DWs will be merged. Completions that belong to the same read request, but have payloads greater than 32 DWs will have one fragment merged up to 32 DWs, in addition to one or more packets containing the rest of the payload (which will also be merged using the same rule). 0h: CMPL_MERGE_UPTO_32DW_INIT (default)
9	R/W	1h	T_PCIE2_RP_TXBA0_CMPL_MERGE_DISABLE: Disables completion merging. By default, completion merging is disabled. 1h: CMPL_MERGE_DISABLE_INIT (default)
8:0	R/W	0h	T_PCIE2_RP_TXBA0_REPLAY_BUF_LIMIT: This field exists per controller. It decides how much of the replay buffer (which is shared among all the controllers in a TMS) is used for the controller. The range of values can be anything from 0 (when controller is disabled) to full depth of the replay buffer size (when there are no other controllers). 0h: REPLAY_BUF_LIMIT_INIT (default)

34.4.8.13 T_PCIE2_RP_TXBA1

This register is specific to the TXBA sub-unit in PCIe 2.0.It controls:

- Replay buffer limits
- Completion merging limits
- Replay timer control fields

Offset: 0xE1C | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:8	R/W	0	T_PCIE2_RP_TXBA1_CMPL_MERGE_THRESHOLD: Completion merging is enabled only if the number of valid completions exceeds this programmed register. This is used to control the bandwidth vs. latency consideration. The maximum meaningful value of this field is the maximum number of outstanding completions (which is also size of header buffer allocated to the controller). 0h: CMPL_MERGE_THRESHOLD_INIT (default)
7:4	R/W	4h	T_PCIE2_RP_TXBA1_CM_OVER_PW_BURST: This field and the PW_OVER_CM_BURST field are input into the downstream arbiter. If there are several downstream posted writes and downstream Completion Merges (CMs), the TXBA arbiter runs PW_OVER_CM_BURST many posted writes before switching to CM. Once the arbiter starts running completions, it runs CM_OVER_PW_BURST completions before switching back to posted writes. 4h: CM_OVER_PW_BURST_INIT (default)
3:0	R/W	4h	T_PCIE2_RP_TXBA1_PW_OVER_CM_BURST: This field and the CM_OVER_PW_BURST field are inputs into the downstream arbiter. If there are several downstream posted writes and downstream CMs, the TXBA arbiter runs PW_OVER_CM_BURST many posted writes before switching to CM. Once the arbiter starts running completions, it runs CM_OVER_PW_BURST completions before switching back to PW. 4h: PW_OVER_CM_BURST_INIT (default)

34.4.8.14 T_PCIE2_RP_FORCEFC

This register updates FC priority flip threshold. Updated FCs become high priority (equivalent priority to UpdateFC timer expiring) if the number of outstanding credits to return for each type exceeds the set threshold.

If the threshold is set to 0, this feature is disabled.

Offset: 0xE20 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	0h	T_PCIE2_RP_FORCEFC_NPD_UNRET_THRESH: 0h: NPD_UNRET_THRESH_INIT (default)

Bit	R/W	Reset	Description
23:16	R/W	0h	T_PCIE2_RP_FORCEFC_NPH_UNRET_THRESH: 0h: NPH_UNRET_THRESH_INIT (default)
15:8	R/W	0h	T_PCIE2_RP_FORCEFC_PWD_UNRET_THRESH: 0h: PWD_UNRET_THRESH_INIT (default)
7:0	R/W	0h	T_PCIE2_RP_FORCEFC_PWH_UNRET_THRESH: 0h: PWH_UNRET_THRESH_INIT (default)

34.4.8.15 T_PCIE2_RP_TIMEOUT0

This is the LINK LTSSM Timeout 0 register.

Note: *The value of this register is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this register.*

Offset: 0xE24 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	14h	T_PCIE2_RP_TIMEOUT0_PAD_SPDCHNG_GEN2: Amount of time to wait for pads to change speed from Gen1 to Gen2. Measured in units of clk25m clock periods. Default value is 7 (~500 ns). Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this register. 14h: PAD_SPDCHNG_GEN2_INIT (default) 7h: PAD_SPDCHNG _GEN2_12MHZ_CLK25M 7h: PAD_SPDCHNG _GEN2_13MHZ_CLK25M 9h: PAD_SPDCHNG _GEN2_16_8MHZ_CLK25M Ah: PAD_SPDCHNG _GEN2_19_2MHZ_CLK25M Dh: PAD_SPDCHNG _GEN2_24MHZ_CLK25M Eh: PAD_SPDCHNG _GEN2_26MHZ_CLK25M 14h: PAD_SPDCHNG_GEN2_38_4MHZ_CLK25M
23:8	R/W	300h	T_PCIE2_RP_TIMEOUT0_PAD_PWRUP_CM: Amount of time to wait for pads to power up (e.g., L1 wake) if common-mode voltage was not maintained during power down (i.e., L1P). Measured in units of clk25m clock periods. Default value is 0xF0 (~20 µs). Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this register. 300h: PAD_PWRUP_CM_INIT (default) F0h: PAD_PWRUP_CM_12MHZ_CLK25M 104h: PAD_PWRUP_CM_13MHZ_CLK25M 150h: PAD_PWRUP_CM_16_8MHZ_CLK25M 180h: PAD_PWRUP_CM_19_2MHZ_CLK25M 1E0h: PAD_PWRUP_CM_24MHZ_CLK25M 208h: PAD_PWRUP_CM_26MHZ_CLK25M 300h: PAD_PWRUP_CM_38_4MHZ_CLK25M
7:0	R/W	14h	T_PCIE2_RP_TIMEOUT0_PAD_PWRUP: Amount of time to wait for pads to power up (e.g., L1 wake) if common-mode voltage was maintained during power down (i.e., L1). Measured in units of clk25m clock periods (83 ns when using a 12 MHz clk25m clock). Default value is 7 (~500 ns). Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this register. 14h: PAD_PWRUP_INIT (default) 7h: PAD_PWRUP_12MHZ_CLK25M 7h: PAD_PWRUP_13MHZ_CLK25M 9h: PAD_PWRUP_16_8MHZ_CLK25M Ah: PAD_PWRUP_19_2MHZ_CLK25M Dh: PAD_PWRUP_24MHZ_CLK25M Eh: PAD_PWRUP_26MHZ_CLK25M 14h: PAD_PWRUP_38_4MHZ_CLK25M

34.4.8.16 T_PCIE2_RP_TIMEOUT1

This is the LINK LTSSM Timeout 1 register.

Offset: 0xE28 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	E8h	T_PCIE2_RP_TIMEOUT1_RCVRY_SPD_UNSUCCESS_EIDLE: Amount of time to wait in E-Idle after an unsuccessful speed change. Measured in units of clk25m clock periods (83 ns when using a 12 MHz clock). Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this field. E8h: RCVRY_SPD_UNSUCCESS_EIDLE_INIT (default) 49h: UNSUCCESS_EIDLE_12MHZ_CLK25M 4Eh: UNSUCCESS_EIDLE_13MHZ_CLK25M 65h: UNSUCCESS_EIDLE_16_8MHZ_CLK25M 74h: UNSUCCESS_EIDLE_19_2MHZ_CLK25M 91h: UNSUCCESS_EIDLE_24MHZ_CLK25M 9Dh: UNSUCCESS_EIDLE_26MHZ_CLK25M E8h: UNSUCCESS_EIDLE_38_4MHZ_CLK25M
23:16	R/W	20h	T_PCIE2_RP_TIMEOUT1_RCVRY_SPD_SUCCESS_EIDLE: Amount of time to wait in E-Idle after a successful speed change. Measured in units of clk25m clock periods. Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this field. 20h: RCVRY_SPD_SUCCESS_EIDLE_INIT (default) Ah: SUCCESS_EIDLE_12MHZ_CLK25M Bh: SUCCESS_EIDLE_13MHZ_CLK25M Eh: SUCCESS_EIDLE_16_8MHZ_CLK25M 10h: SUCCESS_EIDLE_19_2MHZ_CLK25M 14h: SUCCESS_EIDLE_24MHZ_CLK25M 15h: SUCCESS_EIDLE_26MHZ_CLK25M 20h: SUCCESS_EIDLE_38_4MHZ_CLK25M
15:0	R/W	2E8h	T_PCIE2_RP_TIMEOUT1_PAD_SPDCHNG_GEN1: Amount of time to wait for pads to change speed from Gen2 to Gen1. Measured in units of 1 µs. 2E8h: PAD_SPDCHNG_GEN1_INIT (default)

34.4.8.17 T_PCIE2_RP_PRBS

This is the PRBS Results register.

Offset: 0xE34 | Read/Write: R

Bit	R/W	Reset	Description
31:16	R	None	T_PCIE2_RP_PRBS_LOCKED: Each bit indicates that particular lane has locked onto the incoming PRBS pattern and is beginning to count bit errors. For example, if LOCKED[n] is set, lane n has successfully locked onto the incoming PRBS pattern.
15:0	R	None	T_PCIE2_RP_PRBS_ERR_COUNT_OVERFLOW: Each bit indicates number of bit mismatches during PRBS run exceeded 65K for that particular lane. For example, if ERR_COUNT_OVERFLOW[n] is set, lane n registers > 65K bit mismatches.

34.4.8.18 T_PCIE2_RP_PRBS_ERR

This is the PRBS Results register.

Offset: 0xE38 | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0h	T_PCIE2_RP_LANE_PRBS_ERR_COUNT: Number of bit errors detected in the incoming PRBS stream after achieving PRBS lock for the selected lane programmed in ERR_SELECT register.
15:4	R	0	Reserved
3:0	R/W	0h	T_PCIE2_RP_LANE_PRBS_ERR_SELECT: Selects which lane number's Error Count shows up in the ERR_COUNT register 0h: SELECT_INIT (default)

34.4.8.19 T_PCIE2_RP_EIDLE_INFER_TO_0

This is an EIDLE inference timeout register.

Offset: 0xE3C | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	500h	T_PCIE2_RP_EIDLE_INFER_TO_0_RCVRCFG_SUC_SPEED: Infer E-Idle after this amount of time while in Recovery.RcvrCfg or Recovery. Speed and successful_speed_negotiation = 1. Measured in units of UI (Rx symbol times). 500h: RCVRCFG_SUC_SPEED_INIT (default)
15:0	R/W	80h	T_PCIE2_RP_EIDLE_INFER_TO_0_L0_LPBK: Infer E-Idle after this amount of time while in L0 or Loopback. Active Slave. Measured in units of µs. 80h: L0_LPBK_INIT (default)

34.4.8.20 T_PCIE2_RP_EIDLE_INFER_TO_1

This is an EIDLE inference timeout register.

Offset: 0xE40 | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	3E80h	T_PCIE2_RP_EIDLE_INFER_TO_1_UNSUC_SPEED_GEN2: Infer E-Idle after this amount of time while in Recovery. Speed and successful_speed_negotiation = 0 and operating in Gen2 speed. Measured in units of UI (rx symbol times). 3E80h: UNSUC_SPEED_GEN2_INIT (default)
15:0	R/W	7D0h	T_PCIE2_RP_EIDLE_INFER_TO_1_UNSUC_SPEED_GEN1: Infer E-Idle after this amount of time while in Recovery. Speed and successful_speed_negotiation = 0 and operating in Gen1 speed. Measured in units of UI (rx symbol times). 7D0h: UNSUC_SPEED_GEN1_INIT (default)

34.4.8.21 T_PCIE2_RP_LTSSM_DBGREG

This is the LTSSM debug register.

Offset: 0xE44 | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM31: 0h: LINKFSM31_INIT (default) 1h: LINKFSM31_CLEAR
30	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM30: 0h: LINKFSM30_INIT (default) 1h: LINKFSM30_CLEAR
29	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM29: 0h: LINKFSM29_INIT (default) 1h: LINKFSM29_CLEAR
28	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM28: 0h: LINKFSM28_INIT (default) 1h: LINKFSM28_CLEAR
27	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM27: 0h: LINKFSM27_INIT (default) 1h: LINKFSM27_CLEAR
26	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM26: 0h: LINKFSM26_INIT (default) 1h: LINKFSM26_CLEAR
25	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM25: 0h: LINKFSM25_INIT (default) 1h: LINKFSM25_CLEAR
24	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM24: 0h: LINKFSM24_INIT (default) 1h: LINKFSM24_CLEAR
23	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM23: 0h: LINKFSM23_INIT (default) 1h: LINKFSM23_CLEAR
22	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM22: 0h: LINKFSM22_INIT (default) 1h: LINKFSM22_CLEAR

Bit	R/W	Reset	Description
21	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM21: 0h: LINKFSM21_INIT (default) 1h: LINKFSM21_CLEAR
20	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM20: 0h: LINKFSM20_INIT (default) 1h: LINKFSM20_CLEAR
19	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM19: 0h: LINKFSM19_INIT (default) 1h: LINKFSM19_CLEAR
18	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM18: 0h: LINKFSM18_INIT (default) 1h: LINKFSM18_CLEAR
17	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM17: 0h: LINKFSM17_INIT (default) 1h: LINKFSM17_CLEAR
16	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM16: 0h: LINKFSM16_INIT (default) 1h: LINKFSM16_CLEAR
15	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM15: 0h: LINKFSM15_INIT (default) 1h: LINKFSM15_CLEAR
14	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM14: 0h: LINKFSM14_INIT (default) 1h: LINKFSM14_CLEAR
13	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM13: 0h: LINKFSM13_INIT (default) 1h: LINKFSM13_CLEAR
12	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM12: 0h: LINKFSM12_INIT (default) 1h: LINKFSM12_CLEAR
11	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM11: 0h: LINKFSM11_INIT (default) 1h: LINKFSM11_CLEAR
10	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM10: 0h: LINKFSM10_INIT (default) 1h: LINKFSM10_CLEAR
9	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM9: 0h: LINKFSM9_INIT (default) 1h: LINKFSM9_CLEAR
8	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM8: 0h: LINKFSM8_INIT (default) 1h: LINKFSM8_CLEAR
7	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM7: 0h: LINKFSM7_INIT (default) 1h: LINKFSM7_CLEAR
6	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM6: 0h: LINKFSM6_INIT (default) 1h: LINKFSM6_CLEAR
5	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM5: 0h: LINKFSM5_INIT (default) 1h: LINKFSM5_CLEAR
4	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM4: 0h: LINKFSM4_INIT (default) 1h: LINKFSM4_CLEAR
3	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM3: 0h: LINKFSM3_INIT (default) 1h: LINKFSM3_CLEAR
2	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM2: 0h: LINKFSM2_INIT (default) 1h: LINKFSM2_CLEAR

Bit	R/W	Reset	Description
1	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM1: 0h: LINKFSM1_INIT (default) 1h: LINKFSM1_CLEAR
0	RW1C	0h	T_PCIE2_RP_LTSSM_DBGREG_LINKFSM0: 0h: LINKFSM0_INIT (default) 1h: LINKFSM0_CLEAR

34.4.8.22 T_PCIE2_RP_PRIV_ERRSTS

This is the Detailed Private Error status register.

Offset: 0xE48 | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REPLAY_TIMER_EXPIRED_ERR: 0h: REPLAY_TIMER_EXPIRED_ERR_INIT (default) 1h: REPLAY_TIMER_EXPIRED_ERR_CLEAR
30	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_SA_ERR: 0h: SA_ERR_INIT (default) 1h: SA_ERR_CLEAR
29	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_DESKEW_ERR: 0h: DESKEW_ERR_INIT (default) 1h: DESKEW_ERR_CLEAR
28	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_TRAINING_ERR: 0h: TRAINING_ERR_INIT (default) 1h: TRAINING_ERR_CLEAR
27	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_DLLP_CRC_ERR: 0h: DLLP_CRC_ERR_INIT (default) 1h: DLLP_CRC_ERR_CLEAR
26	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_8B10B_ERR: 0h: 8B10B_ERR_INIT (default) 1h: 8B10B_ERR_CLEAR
25	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REPLAY_STARTED_ERR: 0h: REPLAY_STARTED_ERR_INIT (default) 1h: REPLAY_STARTED_ERR_CLEAR
24	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REPLAY_ROLLOVER_ERR: 0h: REPLAY_ROLLOVER_ERR_INIT (default) 1h: REPLAY_ROLLOVER_ERR_CLEAR
23	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_PWH_UPDATE_FC_ERR: 0h: PWH_UPDATE_FC_ERR_INIT (default) 1h: PWH_UPDATE_FC_ERR_CLEAR
22	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_PWH_TOO_MANY_CREDITS_ERR: 0h: PWH_TOO_MANY_CREDITS_ERR_INIT (default) 1h: PWH_TOO_MANY_CREDITS_ERR_CLEAR
21	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_PWD_UPDATE_FC_ERR: 0h: PWD_UPDATE_FC_ERR_INIT (default) 1h: PWD_UPDATE_FC_ERR_CLEAR
20	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_PWD_TOO_MANY_CREDITS_ERR: 0h: PWD_TOO_MANY_CREDITS_ERR_INIT (default) 1h: PWD_TOO_MANY_CREDITS_ERR_CLEAR
19	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_NPH_UPDATE_FC_ERR: 0h: NPH_UPDATE_FC_ERR_INIT (default) 1h: NPH_UPDATE_FC_ERR_CLEAR
18	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_NPH_TOO_MANY_CREDITS_ERR: 0h: NPH_TOO_MANY_CREDITS_ERR_INIT (default) 1h: NPH_TOO_MANY_CREDITS_ERR_CLEAR
17	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_NPD_UPDATE_FC_ERR: 0h: NPD_UPDATE_FC_ERR_INIT (default) 1h: NPD_UPDATE_FC_ERR_CLEAR
16	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_NPD_TOO_MANY_CREDITS_ERR: 0h: NPD_TOO_MANY_CREDITS_ERR_INIT (default) 1h: NPD_TOO_MANY_CREDITS_ERR_CLEAR

Bit	R/W	Reset	Description
15	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_CH_UPDATE_FC_ERR: 0h: CH_UPDATE_FC_ERR_INIT (default) 1h: CH_UPDATE_FC_ERR_CLEAR
14	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_CH_TOO_MANY_CREDITS_ERR: 0h: CH_TOO_MANY_CREDITS_ERR_INIT (default) 1h: CH_TOO_MANY_CREDITS_ERR_CLEAR
13	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_CD_UPDATE_FC_ERR: 0h: CD_UPDATE_FC_ERR_INIT (default) 1h: CD_UPDATE_FC_ERR_CLEAR
12	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_CD_TOO_MANY_CREDITS_ERR: 0h: CD_TOO_MANY_CREDITS_ERR_INIT (default) 1h: CD_TOO_MANY_CREDITS_ERR_CLEAR
11	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_ISOPW_DATA_ERR: 0h: REC_OVFL_ISOPW_DATA_ERR_INIT (default) 1h: REC_OVFL_ISOPW_DATA_ERR_CLEAR
10	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_ISOPW_HDR_ERR: 0h: REC_OVFL_ISOPW_HDR_ERR_INIT (default) 1h: REC_OVFL_ISOPW_HDR_ERR_CLEAR
9	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_ISONP_HDR_ERR: 0h: REC_OVFL_ISONP_HDR_ERR_INIT (default) 1h: REC_OVFL_ISONP_HDR_ERR_CLEAR
8	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_CPL_DATA_ERR: 0h: REC_OVFL_CPL_DATA_ERR_INIT (default) 1h: REC_OVFL_CPL_DATA_ERR_CLEAR
7	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_PW_DATA_ERR: 0h: REC_OVFL_PW_DATA_ERR_INIT (default) 1h: REC_OVFL_PW_DATA_ERR_CLEAR
6	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_NP_DATA_ERR: 0h: REC_OVFL_NP_DATA_ERR_INIT (default) 1h: REC_OVFL_NP_DATA_ERR_CLEAR
5	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_CPL_HDR_ERR: 0h: REC_OVFL_CPL_HDR_ERR_INIT (default) 1h: REC_OVFL_CPL_HDR_ERR_CLEAR
4	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_PW_HDR_ERR: 0h: REC_OVFL_PW_HDR_ERR_INIT (default) 1h: REC_OVFL_PW_HDR_ERR_CLEAR
3	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_NP_HDR_ERR: 0h: REC_OVFL_NP_HDR_ERR_INIT (default) 1h: REC_OVFL_NP_HDR_ERR_CLEAR
2	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_FRAMING_ERR: 0h: FRAMING_ERR_INIT (default) 1h: FRAMING_ERR_CLEAR
1	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_SEQ_ERR: 0h: SEQ_ERR_INIT (default) 1h: SEQ_ERR_CLEAR
0	RW1C	0h	T_PCIE2_RP_PRIV_ERRSTS_LCRC_ERR: 0h: LCRC_ERR_INIT (default) 1h: LCRC_ERR_CLEAR

34.4.8.23 T_PCIE2_RP_PRIV_ERRMSK

This is the Detailed Private Error status register.

Offset: 0xE4C | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_REPLAY_TIMER_EXPIRED_ERR_EN: 1h: REPLAY_TIMER_EXPIRED_ERR_EN_INIT (default) 0h: REPLAY_TIMER_EXPIRED_ERR_EN_OFF 1h: REPLAY_TIMER_EXPIRED_ERR_EN_ON

Bit	R/W	Reset	Description
30	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_SA_ERR_EN: 1h: SA_ERR_EN_INIT (default) 0h: SA_ERR_EN_OFF 1h: SA_ERR_EN_ON
29:28	R/W	0	Reserved
27	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_DLLP_CRC_ERR_EN: 1h: DLLP_CRC_ERR_EN_INIT (default) 0h: DLLP_CRC_ERR_EN_OFF 1h: DLLP_CRC_ERR_EN_ON
26	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_8B10B_ERR_EN: 1h: 8B10B_ERR_EN_INIT (default) 0h: 8B10B_ERR_EN_OFF 1h: 8B10B_ERR_EN_ON
25	R	0	Reserved
24	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_REPLAY_ROLLOVER_ERR_EN: 1h: REPLAY_ROLLOVER_ERR_EN_INIT (default) 0h: REPLAY_ROLLOVER_ERR_EN_OFF 1h: REPLAY_ROLLOVER_ERR_EN_ON
23	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_PWH_UPDATE_FC_ERR_EN: 1h: PWH_UPDATE_FC_ERR_EN_INIT (default) 0h: PWH_UPDATE_FC_ERR_EN_OFF 1h: PWH_UPDATE_FC_ERR_EN_ON
22	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_PWD_TOO_MANY_CREDITS_ERR_EN: 1h: PWD_TOO_MANY_CREDITS_ERR_EN_INIT (default) 0h: PWD_TOO_MANY_CREDITS_ERR_EN_OFF 1h: PWD_TOO_MANY_CREDITS_ERR_EN_ON
21	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_PWD_UPDATE_FC_ERR_EN: 1h: PWD_UPDATE_FC_ERR_EN_INIT (default) 0h: PWD_UPDATE_FC_ERR_EN_OFF 1h: PWD_UPDATE_FC_ERR_EN_ON
20	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_PWD_TOO_MANY_CREDITS_ERR_EN: 1h: PWD_TOO_MANY_CREDITS_ERR_EN_INIT (default) 0h: PWD_TOO_MANY_CREDITS_ERR_EN_OFF 1h: PWD_TOO_MANY_CREDITS_ERR_EN_ON
19	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_NPH_UPDATE_FC_ERR_EN: 1h: NPH_UPDATE_FC_ERR_EN_INIT (default) 0h: NPH_UPDATE_FC_ERR_EN_OFF 1h: NPH_UPDATE_FC_ERR_EN_ON
18	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_NPH_TOO_MANY_CREDITS_ERR_EN: 1h: NPH_TOO_MANY_CREDITS_ERR_EN_INIT (default) 0h: NPH_TOO_MANY_CREDITS_ERR_EN_OFF 1h: NPH_TOO_MANY_CREDITS_ERR_EN_ON
17	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_NPD_UPDATE_FC_ERR_EN: 1h: NPD_UPDATE_FC_ERR_EN_INIT (default) 0h: NPD_UPDATE_FC_ERR_EN_OFF 1h: NPD_UPDATE_FC_ERR_EN_ON
16	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_NPD_TOO_MANY_CREDITS_ERR_EN: 1h: NPD_TOO_MANY_CREDITS_ERR_EN_INIT (default) 0h: NPD_TOO_MANY_CREDITS_ERR_EN_OFF 1h: NPD_TOO_MANY_CREDITS_ERR_EN_ON
15	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_CH_UPDATE_FC_ERR_EN: 1h: CH_UPDATE_FC_ERR_EN_INIT (default) 0h: CH_UPDATE_FC_ERR_EN_OFF 1h: CH_UPDATE_FC_ERR_EN_ON
14	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_CH_TOO_MANY_CREDITS_ERR_EN: 1h: CH_TOO_MANY_CREDITS_ERR_EN_INIT (default) 0h: CH_TOO_MANY_CREDITS_ERR_EN_OFF 1h: CH_TOO_MANY_CREDITS_ERR_EN_ON
13	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_CD_UPDATE_FC_ERR_EN: 1h: CD_UPDATE_FC_ERR_EN_INIT (default) 0h: CD_UPDATE_FC_ERR_EN_OFF 1h: CD_UPDATE_FC_ERR_EN_ON

Bit	R/W	Reset	Description
12	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_CD_TOO_MANY_CREDITS_ERR_EN: 1h: CD_TOO_MANY_CREDITS_ERR_EN_INIT (default) 0h: CD_TOO_MANY_CREDITS_ERR_EN_OFF 1h: CD_TOO_MANY_CREDITS_ERR_EN_ON
11	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_DLL_PROTOCOL_ERR_EN: 1h: DLL_PROTOCOL_ERR_EN_INIT (default) 0h: DLL_PROTOCOL_ERR_EN_OFF 1h: DLL_PROTOCOL_ERR_EN_ON
10:3	R	0	Reserved
2	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_FRAMING_ERR_EN: 1h: FRAMING_ERR_EN_INIT (default) 0h: FRAMING_ERR_EN_OFF 1h: FRAMING_ERR_EN_ON
1	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_SEQ_ERR_EN: 1h: SEQ_ERR_EN_INIT (default) 0h: SEQ_ERR_EN_OFF 1h: SEQ_ERR_EN_ON
0	R/W	1h	T_PCIE2_RP_PRIV_ERRMSK_LCRC_ERR_EN: 1h: LCRC_ERR_EN_INIT (default) 0h: LCRC_ERR_EN_OFF 1h: LCRC_ERR_EN_ON

34.4.8.24 T_PCIE2_RP_LTSSM_TRACE_CONTROL

This is the LTSSM Trace Control register. LTSSM state changes are stored in a RAM. This register controls storing of this change in the RAM.

Offset: 0xE50 | Read/Write: R/W

Bit	R/W	Reset	Description
31:14	R	0	Reserved
13:11	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PRX_LTSSM_MINOR: 0h: TRIG_PRX_LTSSM_MINOR_INIT (default)
10:8	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PTX_LTSSM_MINOR: 0h: TRIG_PTX_LTSSM_MINOR_INIT (default)
7:4	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_LTSSM_MAJOR: 0h: TRIG_LTSSM_MAJOR_INIT (default)
3	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_ON_EVENT: when this bit is set, LTSSM changes will start storing only after its state has reached the state specified in T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_LTSSM_MAJOR, T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PTX_LTSSM_MINOR and T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PRX_LTSSM_MINOR state. T_PCIE2_RP_LTSSM_ 0h: TRIG_ON_EVENT_INIT (default)
2	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_CLEAR_RAM: When written with 1, will clear all entries in RAM. 0h: CLEAR_RAM_INIT (default)
1	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_WRAP_EN: When this bit is set, the RAM is updated every time LTSSM state change happens irrespective of whether the write pointer has wrapped around. When this bit is clear, the RAM update is stopped when the RAM is full. RAM needs to be cleared by writing to T_PCIE2_RP_LTSSM_TRACE_CONTROL_CLEAR_RAM before it can start storing new trace. 0h: WRAP_EN_INIT (default) 0h: WRAP_EN_CLEAR 1h: WRAP_EN_SET
0	R/W	1h	T_PCIE2_RP_LTSSM_TRACE_CONTROL_STORE_EN: This bit controls whether RAM needs to be updated whenever an ltssm_state change happens. 1h: STORE_EN_INIT (default) 0h: STORE_EN_CLEAR 1h: STORE_EN_SET

34.4.8.25 T_PCIE2_RP_LTSSM_TRACE_STATUS

This register helps to check the LTSSM trace which has been stored in RAM.

Offset: 0xE54 | Read/Write: R/W

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21:19	R	None	T_PCIE2_RP_LTSSM_TRACE_STATUS_PRX_LTSSM_MINOR: This field returns the data prx_ltssm_minor in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR
18:16	R	None	T_PCIE2_RP_LTSSM_TRACE_STATUS_PTX_LTSSM_MINOR: This field returns the data ptx_ltssm_minor in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR
15:12	R	None	T_PCIE2_RP_LTSSM_TRACE_STATUS_LTSSM_MAJOR: This field returns the data ltssm_major in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR
11	R	None	T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_DATA_VALID: This field returns the data_valid in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR. Data valid bit is set an entry is written to that location of RAM.
10:6	R/W	0h	T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR: This is the read address to LTSSM trace RAM 0h: READ_ADDR_INIT (default)
5:1	R	None	T_PCIE2_RP_LTSSM_TRACE_STATUS_WRITE_PTR: This field indicates the current location of write pointer. This is especially useful when writing to this RAM is enabled even when RAM is full and write pointer is allowed to wrap around.
0	R	None	T_PCIE2_RP_LTSSM_TRACE_STATUS_RAM_FULL: This field indicates whether all entries in RAM have been written. This is especially useful when writing to this RAM is enabled even when RAM is full and write pointer is allowed to wrap around.

34.4.8.26 T_PCIE2_RP_PG

This register helps power gate the PCIe TMS in NVIDIA-specific ways.

Offset: 0xE58 | Read/Write: R/W

Bit	R/W	Reset	Description
31:3	R	0	Reserved
2	R/W	0h	T_PCIE2_RP_PG_RCVD_PME_TO_ACK_INTR_EN: This field enables generation of interrupt on reception of PME TO ACK TLP. 1h RCVD_PME_TO_ACK_INTR_EN_ENABLED 0h RCVD_PME_TO_ACK_INTR_EN_DISABLED (default)
1	RW1C	0h	T_PCIE2_RP_PG_RCVD_PME_TO_ACK: When the root port receives a PME TO ACK TLP, it sets this field. software can write 1 to this field to clear it to 0. NOTE: this field is reset by Cold Reset 0h RCVD_PME_TO_ACK_INIT (default) 1h RCVD_PME_TO_ACK_CLEAR
0	R/W	0h	T_PCIE2_RP_PG_SEND_PME_TO_MSG: Writing 1 to this field when from previous value of 0 causes the root port to send a PME TO message downstream. When the PME TO message is sent to lower stages of the root port (txtf), this field is automatically cleared to 0 by the root port. 0h SEND_PME_TO_MSG_INIT (default) 1h SEND_PME_TO_MSG_SEND_NOW

34.4.8.27 T_PCIE2_RP_VAR_RANGE0

The PCIe Legacy I/O Decode Range Control Registers allow a programmable I/O address range below 4K to be redirected to a PCIe bus downstream from a RP. This is normally not possible due to the definition of the standard I/O Base/Limit registers in the P2P register set. The purpose is to allow legacy devices on PCIe endpoints to operate at their PC legacy default address ranges.

The range is controlled by a pair of generic 12 bit base/limit address register. The ranges can be mapped anywhere in the lower 4K I/O address space. I/O addresses above 4K can be allocated via standard PCI PnP mechanisms.

Offset: 0xE5C | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R/W	0h	T_PCIE2_RP_VAR_RANGE0_LIMIT: Top address for the variable range positive decode addresses. This will be set to the highest address in the range that the PCIE RP is to decode. Both this range and the base must be set properly for any cycles to be decoded. LIMIT[1:0] always 11'b 0h: LIMIT_DEFAULT (default)
17:16	R	0h	T_PCIE2_RP_VAR_RANGE0_RSVD1: 0h: RSVD1_ZER0 (default)
15:2	R/W	0h	T_PCIE2_RP_VAR_RANGE0_BASE: Base address for the variable range positive decode addresses. This will be set to the lowest address in the range that the PCIE RP is to decode. Both the base and the limit must be set properly for any cycles to be decoded. BASE must be DWORD aligned 0h: BASE_DEFAULT (default)
1	R	0h	T_PCIE2_RP_VAR_RANGE0_RSVD0: 0h: RSVD0_ZER0 (default)
0	R/W	0h	T_PCIE2_RP_VAR_RANGE0_ENABLE: If set then positively decode the variable I/O address range defined in T_PCIE2_VAR_RANGE0. 1h: ENABLE_YES 0h: ENABLE_NO (default)

34.4.8.28 T_PCIE2_RP_VAR_RANGE1

The PCIe Legacy I/O Decode Range Control Registers allow a programmable I/O address range below 4K to be redirected to a PCIe bus downstream from a RP. This is normally not possible due to the definition of the standard I/O Base/Limit registers in the P2P register set. The purpose is to allow legacy devices on PCIe endpoints to operate at their PC legacy default address ranges.

The range is controlled by a pair of generic 12-bit base/limit address register. The ranges can be mapped anywhere in the lower 4K I/O address space. I/O addresses above 4K can be allocated via standard PCI PnP mechanisms.

Offset: 0xE60 | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R/W	0h	T_PCIE2_RP_VAR_RANGE1_LIMIT: Top address for the variable range positive decode addresses. This will be set to the highest address in the range that the PCIe RP is to decode. Both this range and the base must be set properly for any cycles to be decoded. LIMIT[1:0] always 11'b 0h: LIMIT_DEFAULT (default)
17:16	R	0h	T_PCIE2_RP_VAR_RANGE1_RSVD1: 0h: RSVD1_ZER0 (default)
15:2	R/W	0h	T_PCIE2_RP_VAR_RANGE1_BASE: Base address for the variable range positive decode addresses. This will be set to the lowest address in the range that the PCIe RP is to decode. Both the base and the limit must be set properly for any cycles to be decoded. BASE must be DWORD aligned 0h: BASE_DEFAULT (default)
1	R	0h	T_PCIE2_RP_VAR_RANGE1_RSVD0: 0h: RSVD0_ZER0 (default)
0	R/W	0h	T_PCIE2_RP_VAR_RANGE1_ENABLE: If set then positively decode the variable I/O address range defined in T_PCIE2_RP_VAR_RANGE1. 1h: ENABLE_YES 0h: ENABLE_NO (default)

34.4.8.29 T_PCIE2_RP_ECTL_1_R1

Offset: 0xE80 | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved

Bit	R/W	Reset	Description
27:24	R/W	0h	T_PCIE2_RP_ECTL_1_R1_RX_FELS_1C: 0h: RX_FELS_1C_DEFAULT (default)
23:20	R	0	Reserved
19:18	R/W	0h	T_PCIE2_ RP_ECTL_1_R1_RX_TERM_CTRL_1C: 0h: RX_TERM_CTRL_1C_DEFAULT (default)
17:16	R/W	0h	T_PCIE2_RP_ECTL_1_R1_TX_TERM_CTRL_1C: 0h: TX_TERM_CTRL_1C_DEFAULT (default)
15:12	R/W	0h	T_PCIE2_RP_ECTL_1_R1_TX_DRV_CTRL_1C: 0h: TX_DRV_CTRL_1C_DEFAULT
11:8	R/W	0h	T_PCIE2_RP_ECTL_1_R1_TX_DRV_SLEW_1C: 0h: TX_DRV_SLEW_1C_DEFAULT (default)
7:6	R	0	Reserved
5:0	R/W	1Fh	T_PCIE2_RP_ECTL_1_R1_TX_DRV_AMP_1C: 1Fh: TX_DRV_AMP_1C_DEFAULT (default)

34.4.8.30 T_PCIE2_RP_ECTL_2_R1

Offset: 0xE84 | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R	0	Reserved
19:16	R/W	0h	T_PCIE2_RP_ECTL_2_R1_RX_IQ_CTRL_1C: 0h: R1_RX_IQ_CTRL_1C_DEFAULT
15:0	R/W	Fh	T_PCIE2_RP_ECTL_2_R1_RX_CTLE_1C: Fh: R1_RX_CTLE_1C_DEFAULT

34.4.8.31 T_PCIE2_RP_ECTL_3_R1

Offset: 0xE88 | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	T_PCIE2_RP_ECTL_3_R1_RX_DFE_1C: 0h: R1_RX_DFE_1C_DEFAULT

34.4.8.32 T_PCIE2_RP_ECTL_4_R1

Offset: 0xE8C | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	67h	T_PCIE2_RP_ECTL_4_R1_RX_CDR_CTRL_1C: 67h: R1_RX_CDR_CTRL_1C_DEFAULT
15:8	R	0	Reserved
7:0	R/W	0h	T_PCIE2_RP_ECTL_4_R1_RX_PI_CTRL_1C: 0h: R1_RX_PI_CTRL_1C_DEFAULT

34.4.8.33 T_PCIE2_RP_ECTL_5_R1

Offset: 0xE90 | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	T_PCIE2_RP_ECTL_5_R1_RX_EQ_CTRL_L_1C: 0h: R1_RX_EQ_CTRL_L_1C_DEFAULT

34.4.8.34 T_PCIE2_RP_ECTL_6_R1

Offset: 0xE94 | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	T_PCIE2_RP_ECTL_6_R1_RX_EQ_CTRL_H_1C: 0h: R1_RX_EQ_CTRL_H_1C_DEFAULT

34.4.8.35 T_PCIE2_RP_ECTL_7_R1

Offset: 0xE98 | Read/Write: R/W

Bit	R/W	Reset	Description
31:14	R	0	Reserved
13:8	R/W	0h	T_PCIE2_RP_ECTL_7_R1_TX_DRV_PRE_1C 0h: R1_TX_DRV_PRE_1C_DEFAULT
7:6	R	0	Reserved
5:0	R/W	Ah	T_PCIE2_RP_ECTL_7_R1_TX_DRV_POST_1C: Ah: R1_TX_DRV_POST_1C_DEFAULT

34.4.8.36 T_PCIE2_RP_ECTL_1_R2

Offset: 0xEA0 | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:24	R/W	0h	T_PCIE2_RP_ECTL_1_R2_RX_FELS_1C: 0h: RX_FELS_1C_DEFAULT (default)
23:20	R	0	Reserved
19:18	R/W	0h	T_PCIE2_RP_ECTL_1_R2_RX_TERM_CTRL_1C: 0h: RX_TERM_CTRL_1C_DEFAULT (default)
17:16	R/W	0h	T_PCIE2_RP_ECTL_1_R2_TX_TERM_CTRL_1C: 0h: TX_TERM_CTRL_1C_DEFAULT (default)
15:12	R/W	0h	T_PCIE2_RP_ECTL_1_R2_TX_DRV_CTRL_1C: 0h: TX_DRV_CTRL_1C_DEFAULT (default)
11:8	R/W	0h	T_PCIE2_RP_ECTL_1_R2_TX_DRV_SLEW_1C: 0h: TX_DRV_SLEW_1C_DEFAULT (default)
7:6	R	0	Reserved
5:0	R/W	1Fh	T_PCIE2_RP_ECTL_1_R2_TX_DRV_AMP_1C: 1Fh: TX_DRV_AMP_1C_DEFAULT (default)

34.4.8.37 T_PCIE2_RP_ECTL_2_R2

Offset: 0xEA4 | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R	0	Reserved
19:16	R/W	0h	T_PCIE2_RP_ECTL_2_R2_RX_IQ_CTRL_1C: 0h: RX_IQ_CTRL_1C_DEFAULT (default)
15:0	R/W	8Fh	T_PCIE2_RP_ECTL_2_R2_RX_CTLE_1C: 8Fh: RX_CTLE_1C_DEFAULT (default)

34.4.8.38 T_PCIE2_RP_ECTL_3_R2

Offset: 0xEA8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	T_ PCIE2_RP_ECTL_3_R2_RX_DFE_1C: 0h: R2_RX_DFE_1C_DEFAULT

34.4.8.39 T_PCIE2_RP_ECTL_4_R2

Offset: 0xEAC | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	C7h	T_PCIE2_RP_ECTL_4_R2_RX_CDR_CTRL_1C: C7h: R2_RX_CDR_CTRL_1C_DEFAULT
15:8	R	0	Reserved
7:0	R/W	0h	T_PCIE2_RP_ECTL_4_R2_RX_PI_CTRL_1C: 0h: R2_RX_PI_CTRL_1C_DEFAULT

34.4.8.40 T_PCIE2_RP_ECTL_5_R2

Offset: 0xEB0 | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	T_PCIE2_RP_ECTL_5_R2_RX_EQ_CTRL_L_1C: 0h: R2_RX_EQ_CTRL_L_1C_DEFAULT

34.4.8.41 T_PCIE2_RP_ECTL_6_R2

Offset: 0xEB4 | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	T_PCIE2_RP_ECTL_6_R2_RX_EQ_CTRL_H_1C: 0h: R2_RX_EQ_CTRL_H_1C_DEFAULT

34.4.8.42 T_PCIE2_RP_ECTL_7_R2

Offset: 0xEB8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29:24	R/W	0h	T_PCIE2_RP_ECTL_7_R2_TX_DRV_PRE_SEL1_1C: 0h TX_DRV_PRE_SEL1_1C_DEFAULT (default)
23:22	R	0	Reserved
21:16	R/W	Ah	T_PCIE2_RP_ECTL_7_R2_TX_DRV_POST_SEL1_1C: Ah TX_DRV_POST_SEL1_1C_DEFAULT (default)
15:14	R	0	Reserved
13:8	R/W	0h	T_PCIE2_RP_ECTL_7_R2_TX_DRV_PRE_SEL0_1C 0h: R2_TX_DRV_PRE_SEL0_1C_DEFAULT
7:6	R	0	Reserved
5:0	R/W	Fh	T_PCIE2_RP_ECTL_7_R2_TX_DRV_POST_SEL0_1C: Fh: R2_TX_DRV_POST_SEL0_1C_DEFAULT

34.4.8.43 T_PCIE2_RP_VEND_XP

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF00 | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	T_PCIE2_RP_VEND_XP_FORCE_COMPLIANCE: 0h: FORCE_COMPLIANCE_INIT (default)
30	R	None	T_PCIE2_RP_VEND_XP_DL_UP: This read-only bit tells status of Data Link Layer in XP. This bit was added before the DL Link Active Reporting feature was introduced in the PCI Express Specification. It asserts when the DL enters the InitFC2 phase of training. This means that DL_UP will assert before the DL_LINK_ACTIVE bit asserts.

Bit	R/W	Reset	Description
29	R/W	0h	T_PCIE2_RP_VEND_XP_INTERLEAVE_DLLPS: Setting this bit will cause the DL to schedule Ack and UpdateFC packets earlier or later than their respective timer expiration times in order to decrease the likelihood of losing efficiency by sending two DLLPs consecutively. 0h: INTERLEAVE_DLLPS_INIT (default)
28	R/W	0h	T_PCIE2_RP_VEND_XP_OPPORTUNISTIC_UPDATEFC: If this bit is set, the DL will send any pending UpdateFC packet whenever it has nothing else to send, instead of waiting for the UpdateFC timer to expire. 0h: OPPORTUNISTIC_UPDATEFC_INIT (default)
27	R/W	0h	T_PCIE2_RP_VEND_XP_OPPORTUNISTIC_ACK: If this bit is set, the DL will send pending Acks whenever it has nothing else to send, instead of waiting for the Ack Timer to expire. 0h: OPPORTUNISTIC_ACK_INIT (default)
26	R/W	0h	T_PCIE2_RP_VEND_XP_TRAIN_ERR_ENABLE: Enables reporting of training errors. 0h: TRAIN_ERR_ENABLE_INIT (default)
25:18	R/W	0h	T_PCIE2_RP_VEND_XP_UPDATE_FC_THRESHOLD: This field specifies an override for the UpdateFC frequency. Setting this field to a non-zero value will cause the TL to use the programmed value MULTIPLIED BY TWO as the UpdateFC timer limit instead of the default value which is based on the negotiated width as described in section 2.6.1.2 of the PCI Express Base Specification, Rev 1.0a. The UpdateFC timer limit corresponds to the UpdateFC Transmission Latency Limit listed in table 2-28 of the specification, except that it should *not* include the InternalDelay. 0h: UPDATE_FC_THRESHOLD_INIT (default)
17:2	R	None	T_PCIE2_RP_VEND_XP_PRBS_STAT: This field returns the results of loopback mode testing. Each bit represents the status of one lane (1 == PASS).
1	R/W	0h	T_PCIE2_RP_VEND_XP_PRBS_EN: Enables the root port as loopback master using PRBS-23 as the test pattern. 0h: PRBS_EN_DISABLED (default) 1h: PRBS_EN_ENABLED
0	R/W	0h	T_PCIE2_RP_VEND_XP_EMULATION: Enables emulation mode operation. 0h: EMULATION_OFF (default) 1h: EMULATION_ON

34.4.8.44 T_PCIE2_RP_VEND_XP1

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF04 | Read/Write: R/W

Bit	R/W	Reset	Description
31:29	R/W	2h	T_PCIE2_RP_VEND_XP1_L1_EXIT_LATENCY: This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L1 to L0. The value is reflected in the LINK_CAPABILITIES register. Defined encodings are: 000b: Less than 1 µs 001b: 1 µs to less than 2 µs 010b: 2 µs to less than 4 µs 011b: 4 µs to less than 8 µs 100b: 8 µs to less than 16 µs 101b: 16 µs to less than 32 µs 110b: 32 µs-64 µs 111b: More than 64 µs 2h: L1_EXIT_LATENCY_INIT (default) 0h: L1_EXIT_LATENCY_LT_1 1h: L1_EXIT_LATENCY_LT_2 2h: L1_EXIT_LATENCY_LT_4 3h: L1_EXIT_LATENCY_LT_8 7h: L1_EXIT_LATENCY_GT_64

Bit	R/W	Reset	Description
28	R/W	0h	T_PCIE2_RP_VEND_XP1_FORCE_DOWNSTREAM_NO_SNOOP: When this bit is set and ENABLE_NO_SNOOP bit is also set, NO_SNOOP bit is set on all downstream TLPs (except for I/O and Config, for which NO_SNOOP is never set) 1h: FORCE_DOWNSTREAM_NO_SNOOP_YES 0h: FORCE_DOWNSTREAM_NO_SNOOP_NO (default)
27	R/W	0h	T_PCIE2_RP_VEND_XP1_FORCE_UPSTREAM_NONCOH: When set this bit causes all upstream memory traffic (except MSI) to be non-coherent. 0h: FORCE_UPSTREAM_NONCOH_INIT (default) 1h: FORCE_UPSTREAM_NONCOH_ENABLED 0h: FORCE_UPSTREAM_NONCOH_DISABLED
26:19	R/W	0h	T_PCIE2_RP_VEND_XP1_LINK_PVT_CTL: bit[0] : ACK_L1_NO_WAIT Enable behavior described in the Errata C7 of the PCI Express Base Specification, v1.0a, 7 October 2003. Software should always set this bit to 1. When this bit is set to 1 on an upstream component, that component will not wait to accumulate the minimum number of credits required to send the largest possible packet for any FC type before sending PM_Request_Ack DLLPs downstream. After receiving a PM_Enter_L1 DLLP, it will begin sending PM_Request_Ack DLLPs as soon as it has received acknowledgment for the last TLP that has been sent. bit [1]: L23_READY_NO_D3 Enable entry into the L2/3 Ready state when the component is not in the D3 PMCSR state. If this bit is set to 0, the component will not allow the link to enter the L2/3 Ready state if it is not in D3. If the bit is set to 1, the link will enter L2/3 Ready after the component receives a PME_Turn_Off message, regardless of the PMCSR state. bit [2]: L1_ASPM_SUPPORT This bit determines the reported value of L1 ASPM support in the link capability register (ACTIVE_STATE_LINK_PM_SUPPORT, bit 11). bit [3]: DONT_MERGE_PMASNAK With the default setting for this bit (0), a group of PM_Active_State_Request_L1 DLLPs from the endpoint will cause a single PM_ASPM_Nak TLP to be sent. If this bit is set to 1, multiple PM_ASPM_Nak TLP messages will be sent, possibly as many as one PM_ASPM_Nak TLP for each PM_Active_State_Request_L1 DLLP. bit [4]: IGNORE_L0S This bit overrides the L0s setting for the Active Sate PM Control register. If bit[4] == 0 (do not override), then bit 0 of the T_PCIE2_RP_LINK_CONTROL_STATUS register is used to determine if the Tx side drivers should enter L0s or not. If bit[4] == 1 (override), then bit 0 of T_PCIE2_RP_LINK_CONTROL_STATUS is ignored, and Tx.L0s is not entered. bit [7]: NP_REQ_TIMEOUT If set, disable downstream NP request timeout. This is for debug purposes to hang instead of timeout. 0h: LINK_PVT_CTL_INIT (default)
18:10	R/W	0h	T_PCIE2_RP_VEND_XP1_ACK_TIMER_LIMIT: This field overrides the Ack Timer Limit for this root port. Setting this field to a non-zero value will cause the DL to use the programmed value for the Ack Timer Limit instead of the default value, which is based on the negotiated width as described in section 3.5.3.1 of the PCI Express Base Specification, Rev 1.0a. The Ack Timer Limit corresponds to the Unadjusted Ack Transmission Latency Limit listed in table 3-5 of the specification, except that it should include the Tx_L0s_Adjustment, and should *not* include the InternalDelay. 0h: ACK_TIMER_LIMIT_INIT (default)
9	R	0	Reserved
8	R/W	1h	T_PCIE2_RP_VEND_XP1_RNCTRL_GEN2_WAIT_FOR_FIRST_EIES: When this bit is programmed to 1, in the Config.LinkWidth.Start state, the inactive lanes that are activated will wait for the next EIEOS transmission before sending the first packet. If programmed to 0, it will not wait for the next EIEOS transmission before sending the first packet. 1h: RNCTRL_GEN2_WAIT_FOR_FIRST_EIES_INIT (default)
7	R	0h	T_PCIE2_RP_VEND_XP1_RNCTRL_EN: When this bit is programmed to a 1, it triggers the dynamic link width re-negotiation procedure in XP. This bit is a constant and always returns a value of 0 on a read. 0h: RNCTRL_EN_ZERO (default)
6	R/W	1h	T_PCIE2_RP_VEND_XP1_RNCTRL_GEN2_LINK_UPGRADE: When this bit is programmed to 1, it selects the 2.0-compliant link width upgrade protocol. If programmed to 0, it selects the NV-proprietary link width upgrade protocol. 1h: RNCTRL_GEN2_LINK_UPGRADE_INIT (default)
5:0	R/W	10h	T_PCIE2_RP_VEND_XP1_RNCTRL_MAXWIDTH: This field indicates the maximum link width required at the end of dynamic link width re-negotiation process. The default value is 16. 10h: RNCTRL_MAXWIDTH_INIT (default)

34.4.8.45 T_PCIE2_RP_VEND_XP2

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF08 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	0h	T_PCIE2_RP_VEND_XP2_L0S_UPDATE_WAKE: Setting this field to a non-zero value will cause the DL to wake the PL out of L0s in advance of sending it an UpdateFC packet. When the DL sees that there are fewer cycles left in the UpdateFC timer than the value in the L0S_UPDATE_WAKE register, it will tell the PL to wake out of L0s. 0h: L0S_UPDATE_WAKE_INIT (default)
23:18	R	0	Reserved
17:8	R/W	3FFh	T_PCIE2_RP_VEND_XP2_L0S_THRESHOLD: This field controls the idle time required for TxL0s entry from L0, in symbol times (250 MHz clocks). Once the XP detects that it has no more DLLPs/TLPs to send, it will insert precisely (L0S_THRESHOLD + 1) logical idles between the END symbol of the last DLLP/TLP and the COM of the IDL Ordered Set. If L0S_THRESHOLD == 0x3FF, the L0s entry latency is instead (endpoint N_FTS)*4 symbol times, where (endpoint N_FTS) is the N_FTS value advertised by the endpoint during training. Therefore, the threshold can be set to anywhere from 1 to 1023 symbol times (4ns to 4092ns). 3FFh: L0S_THRESHOLD_INIT (default) 3FFh: L0S_THRESHOLD_REMOTE_NFTS
7:0	R/W	0h	T_PCIE2_RP_VEND_XP2_L0S_ACK_WAKE: Setting this field to a non-zero value will cause the DL to wake the PL out of L0s in advance of sending it an Ack packet. When the DL sees that there are fewer cycles left in the Ack timer than the value in the L0S_ACK_WAKE register, it will tell the PL to awaken from L0s. 0h: L0S_ACK_WAKE_INIT (default)

34.4.8.46 T_PCIE2_RP_VEND_XV_TIMEOUT

XP Extended Control Register 2

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF0C | Read/Write: R/W | Reset: 0x030000fa (0b000000110000xxxxxxxxxx0011111010)

Bit	Reset	Description
31:20	N333	N100MS_DFPCI: 24 = N250 48 = N333
19:10	0	Reserved
9:0	DDEFAULT	MICROSECOND: 250 = DDEFAULT 250 = N250 278 = N278 313 = N313 357 = N357 417 = N417 500 = N500 555 = N555

34.4.8.47 T_PCIE2_RP_VEND_SLOT_STRAP

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

This is a back-door register used to set various bits in the Slot Capabilities Register, which is part of the PCI-Express Capability Structure, as defined by the PCI-Express Specification. The bits listed here map 1:1 with the bits in the Slot Capabilities Register.

Note: *Bits marked "Reserved" may actually be writable in some chips. Therefore writes to this register must not change the default POR values for the bits marked "Reserved".*

Offset: 0xF20 | Read/Write: R/W

Bit	R/W	Reset	Description
31:19	R/W	0h	T_PCIE2_RP_VEND_SLOT_STRAP_PHYSICAL_SLOT_NUMBER: The physical numbering of the slots is left up to the board designer. The board designer must communicate the physical slot numbers to the SBIOS developer as well. 0h: PHYSICAL_SLOT_NUMBER_INIT (default)
18:17	R	0	Reserved
16:15	R/W	0h	T_PCIE2_RP_VEND_SLOT_STRAP_SLOT_POWER_LIMIT_SCALE: This field must be programmed according to the unique power-rail capabilities of each board design. Board designers are responsible for communicating the power limits of each slot to the SBIOS developers. 0h: SLOT_POWER_LIMIT_SCALE_INIT (default)
14:7	R/W	0h	T_PCIE2_RP_VEND_SLOT_STRAP_SLOT_POWER_LIMIT_VALUE: This field must be programmed according to the unique power-rail capabilities of each board design. Board designers are responsible for communicating the power limits of each slot to the SBIOS developers. 0h: SLOT_POWER_LIMIT_VALUE_INIT (default)
6:0	R	0	Reserved

34.4.8.48 T_PCIE2_RP_XP_REF

This register contains diagnostic bits used in XP.

Offset: 0xF30 | Read/Write: R/W

Bit	Reset	Description
31:14	0x01D4C	CPL_TO_CUSTOM_VALUE: User-specified value to use for completion timeout. Program this register to the desired completion timeout value in ns divided by16 divided by the clock period of clk25m (83.33 ns). Default value is 0x1d4c, which works out to be 10 ms. This value has meaning only when CPL_TO_OVERRIDE is enabled. 0x0001D4C: INIT
13	0	CPL_TO_OVERRIDE: Enables the user override mode for completion timeout. Default is disabled. 0: INIT 1: ENABLE 0: DISABLE
12	1	ADVANCE_BY_2_CYA: Diagnostic bit. Default is enabled. 1: INIT 1: ENABLE 0: DISABLE
11	1	NAK_SCHEDULE_CYA: Diagnostic bit. Default is enabled. 1: INIT 1: ENABLE 0: DISABLE
10	1	RXL2LINK_NPT_EMPTY_CYA: Diagnostic bit. Default is enabled. 1: INIT 1: ENABLE 0: DISABLE
9	1	L2_READ_FIX_EN: Diagnostic bit. When enabled, Downstream FPCI transactions are retried in Link Config state. 1: INIT 1: ENABLE 0: DISABLE
8	0	MICROSECOND_ENABLE: When enabled, use the counter value from REF_MICROSECOND_LIMIT. When disabled, use the hard-coded value. 0: INIT 1: ENABLE 0: DISABLE

Bit	Reset	Description
7:0	0x27	MICROSECOND_LIMIT: Counter value to be used to reset the one microsecond counter used for UpdateFC scheduling. Note: This value is correct only for the default frequency of clk25m. If the clk25m frequency is changed, then software is responsible for changing the value of this register. 0x27: INIT 0xC: 12MHZ_CLK25M 0xD: 13MHZ_CLK25M 0x11: 16_8MHZ_CLK25M 0x14: 19_2MHZ_CLK25M 0x18: 24MHZ_CLK25M 0x1A: 26MHZ_CLK25M 0x27: 38_4MHZ_CLK25M

34.4.8.49 T_PCIE2_RP_VEND_CTL1

This register contains fields unique to NVIDIA's implementation of PCI Express devices. This is a control register for general feature and function control.

Offset: 0xF48 | Read/Write: R/W

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21	R/W	1h	T_PCIE2_RP_VEND_CTL1_POLLING_RESET_FIX_EN: 1h: POLLING_RESET_FIX_EN_INIT (default) 1h: POLLING_RESET_FIX_EN_ENABLE 0h: POLLING_RESET_FIX_EN_DISABLE
20	R/W	1h	T_PCIE2_RP_VEND_CTL1_HOTPLUG_IN_TRAFFIC_EN: 1h: HOTPLUG_IN_TRAFFIC_EN_INIT (default) 1h: HOTPLUG_IN_TRAFFIC_EN_ENABLE 0h: HOTPLUG_IN_TRAFFIC_EN_DISABLE
19	R/W	0h	T_PCIE2_RP_VEND_CTL1_NISO2ISO: When set to 1, all upstream non-ISO-PW and non-ISO-NP will be upgraded to ISO. 0h: NISO2ISO_INIT (default) 1h: NISO2ISO_ENABLE 0h: NISO2ISO_DISABLE
18	R/W	0h	T_PCIE2_RP_VEND_CTL1_ALLOW_UPSTREAM_CMPL_OVERTAKE_PW: When set to 1, the UBFI allows the CPL to bypass posted writes regardless of the RO bit 0h: ALLOW_UPSTREAM_CMPL_OVERTAKE_PW_DIS (default) 1h: ALLOW_UPSTREAM_CMPL_OVERTAKE_PW_EN
17	R/W	0h	T_PCIE2_RP_VEND_CTL1_SPLIT_ARB_BLOCK_COH: When set to 0, take the single ufpci2fpci_arb_block_coherent as input. When set to 1, take the separate inputs of ufa2fpci_arb_block_coherent_pw and ufa2fpci_arb_block_coherent_np. 0h: SPLIT_ARB_BLOCK_COH_INIT (default) 0h: SPLIT_ARB_BLOCK_COH_DIS 1h: SPLIT_ARB_BLOCK_COH_EN
16	R/W	0h	T_PCIE2_RP_VEND_CTL1_HIDE_MSIMAP: 0h: HIDE_MSIMAP_DIS (default) 1h: HIDE_MSIMAP_EN
15	R/W	1h	T_PCIE2_RP_VEND_CTL1_LINKACTV_REPORTING: This is a backdoor bit for setting the Data Link Layer Link Active Reporting Capable bit (bit 20 of the Link Capabilities Register). This bit *MUST* be set whenever hot-plug is enabled. Otherwise, it can be set as desired. 1h: LINKACTV_REPORTING_CAPABLE (default) 0h: LINKACTV_REPORTING_NOT_CAPABLE
14	R/W	1h	T_PCIE2_RP_VEND_CTL1_P2P_ISO2NISO: If set, forces upstream peer-to-peer ISO requests to be peer-to-peer NONISO. 1h: P2P_ISO2NISO_EN (default) 0h: P2P_ISO2NISO_DIS
13	R/W	0h	T_PCIE2_RP_VEND_CTL1_ERPT: This bit, when 0, hides the entire AER Capability structure from the capabilities list. It causes the "next pointer" of the previous capability to point to NULL. Normally, AER is the last capability in the list. 1h: ERPT_EN 0h: ERPT_DIS (default)

Bit	R/W	Reset	Description
12	R/W	0h	T_PCIE2_RP_VEND_CTL1_HIDE_MSI_CAP: This bit, when 1, hides the entire MSI Capability structure from the capabilities list. It causes the "next pointer" of the previous capability to point to the capability that comes after MSI. 0h: HIDE_MSI_CAP_DIS (default) 1h: HIDE_MSI_CAP_EN
11:5	R	0	Reserved
4	R/W	0h	T_PCIE2_RP_VEND_CTL1_ACCEPT_MSGD1: When set, allows acceptance of NVIDIA specific vendor type message with data. 0h: ACCEPT_MSGD1_DIS (default) 1h: ACCEPT_MSGD1_EN
3:1	R/W	7h	T_PCIE2_RP_VEND_CTL1_TC2ISO_MAP: This field specifies TC2ISO_MAP[2:0]. Upstream traffic with TC[2:0] >= TC2ISO_MAP[2:0] are sent as ISO requests on UFPCI Since TC[2:0] == 0 can never be sent to ISO channel, TC2ISO_MAP[2:0] == 0 is used to force ALL traffic to NONISO 7h: TC2ISO_MAP_7 (default)
0	R/W	0h	T_PCIE2_RP_VEND_CTL1_P2P_BLOCK_P2P_ONLY: When set, ignores ufa2fpci_arb_block_coherent for upstream peer2peer requests. 1h: P2P_BLOCK_P2P_ONLY_EN 0h: P2P_BLOCK_P2P_ONLY_DIS (default)

34.4.8.50 T_PCIE2_RP_VEND_XP_BIST

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

IOBIST and Characterization Control Register

Offset: 0xF4C | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	0	Reserved
30	R/W	0h	T_PCIE2_RP_VEND_XP_BIST_GOTO_DETECT_ON_SURPRISE_EIDLE: 0h: GOTO_DETECT_ON_SURPRISE_EIDLE_INIT (default)
29	R/W	0h	T_PCIE2_RP_VEND_XP_BIST_ENABLE_SERR_REPORTING: 0h: ENABLE_SERR_REPORTING_INIT (default)
28	R/W	1h	T_PCIE2_RP_VEND_XP_BIST_GOTO_L1_L2_AFTER_DLLP_DONE: 1h: GOTO_L1_L2_AFTER_DLLP_DONE_INIT (default)
27	R/W	0h	T_PCIE2_RP_VEND_XP_BIST_CTRL_IGNORE_LPBK_EXIT: 0h: CTRL_IGNORE_LPBK_EXIT_INIT (default)
26	R/W	0h	T_PCIE2_RP_VEND_XP_BIST_CTRL_RELAX_LPBK_ENTRY: 0h: CTRL_RELAX_LPBK_ENTRY_INIT (default)
25	R/W	0h	T_PCIE2_RP_VEND_XP_XP_BIST_CTRL_FORCE_PRBS_RST: 0h: INIT (default)
24	R/W	0h	T_PCIE2_RP_VEND_XP_FORCE_DEEMPHASIS_ADVERTISED_EN: 0h: INIT (default)
23	R	0h	Reserved
22	R/W	0h	T_PCIE2_RP_VEND_XP_FORCE_RECEIVER_COMPLIANCE_EN: 0h: INIT (default)
21	R/W	0h	T_PCIE2_RP_VEND_XP_FORCE_COMPLIANCE_GEN2_SPEED_EN: 0h: INIT (default)
20	R/W	0h	T_PCIE2_RP_VEND_XP_BIST_FORCE_COMPLIANCE_AND_ADVERTISE_MODE: 0h: FORCE_COMPLIANCE_AND_ADVERTISE_MODE_INIT (default)
19:0	R	0	Reserved

34.4.8.51 T_PCIE2_RP_VEND_XP_FTS

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF54 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	80h	T_PCIE2_RP_VEND_XP_FTS_TS_DETECT_START: 80h: TS_DETECT_START_INIT (default)
23:16	R/W	40h	T_PCIE2_RP_VEND_XP_FTS_FTS_DETECT_START: This field represents the number of symbol times to wait before the root port begins looking at FTS ordered sets when exiting L0s -- the incoming data is essentially ignored during this time. 40h: FTS_DETECT_START_INIT (default)
15:8	R	None	T_PCIE2_RP_VEND_XP_FTS_N_FTS_REMOTE: NV_XVE_PRIV_XP_N_FTS_REMOTE represents the N_FTS value advertised by the remote device as recorded from the N_FTS symbol of the received TS ordered sets.
7:0	R/W	1Fh	T_PCIE2_RP_VEND_XP_FTS_N_FTS: This field represents the N_FTS value to advertise to the device on the other side of the link. NOTE: If this field is modified from its default (POR) value, the Replay Timer Limit in T_PCIE2_RP_VEND_XP1 must be adjusted accordingly. 1Fh: N_FTS_INIT (default)

34.4.8.52 T_PCIE2_RP_VEND_XP_STATS0

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF58 | Read/Write: R/W

Bit	R/W	Reset	Description
31:29	R	0	Reserved
28	R/W	0h	T_PCIE2_RP_VEND_XP_STATS0_FAILED_L0S_EXITS_INF: When set to 1, the failed L0s exits counter accumulates indefinitely (i.e. the one millisecond timer is ignored). 0h: FAILED_L0S_EXITS_INF_INIT (default)
27:25	R	0	Reserved
24	R	0h	T_PCIE2_RP_VEND_XP_STATS0_FAILED_L0S_EXITS_LC: Loads the current value from the failed L0s exits counter into the VEND_XP_STATS1 register, and then clears the failed L0s exits counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. 0h: FAILED_L0S_EXITS_LC_INIT (default)
23:21	R	0	Reserved
20	R/W	0h	T_PCIE2_RP_VEND_XP_STATS0_NAKS_RCVD_INF: When set to 1, the NAKs received counter accumulates indefinitely (i.e. the 64 microsecond timer is ignored). 0h: NAKS_RCVD_INF_INIT (default)
19:17	R	0	Reserved
16	R	0h	T_PCIE2_RP_VEND_XP_STATS0_NAKS_RCVD_LC: Loads the current value from the NAKs received counter into the VEND_XP_STATS1 register, and then clears the NAKs received counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. 0h: NAKS_RCVD_LC_INIT (default)
15:13	R	0	Reserved
12	R/W	0h	T_PCIE2_RP_VEND_XP_STATS0_CRC_ERRORS_INF: When set to 1, the CRC error counter accumulates indefinitely (i.e. the 64 microsecond timer is ignored). 0h: CRC_ERRORS_INF_INIT (default)
11:9	R	0	Reserved
8	R	0h	T_PCIE2_RP_VEND_XP_STATS0_CRC_ERRORS_LC: Loads the current value from the CRC error counter into the VEND_XP_STATS1 register, and then clears the CRC error counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. 0h: CRC_ERRORS_LC_INIT (default)
7:5	R	0	Reserved
4	R/W	0h	T_PCIE2_RP_VEND_XP_STATS0_8B10B_ERRORS_INF: When set to 1, the 8b/10b error counter accumulates indefinitely (i.e. the one microsecond timer is ignored). 0h: 8B10B_ERRORS_INF_INIT (default)

Bit	R/W	Reset	Description
3:1	R	0	Reserved
0	R	0h	T_PCIE2_RP_VEND_XP_STATS0_8B10B_ERRORS_LC: Loads the current value from the 8B/10B error counter into the VEND_XP_STATS1 register, and then clears the 8b/10b error counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. Note: This counter will count the number of 8b10b errors in a time window of size 14 clk25m clock periods from the time a 1 is written into this field. Depending on the clock frequency of clk25m, this window will be sized differently. Default is 12 clock periods (1 us). 0h: 8B10B_ERRORS_LC_INIT (default)

34.4.8.53 T_PCIE2_RP_VEND_XP_STATS1

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF5C | Read/Write: R

Bit	R/W	Reset	Description
31:24	R	None	T_PCIE2_RP_VEND_XP_STATS1_FAILED_L0S_EXITS: Counts the number of times the RX side of the XP went into Recovery in one millisecond, due to a failed attempt to exit L0s. Saturates at 255 (counter does not roll over to 0).
23:16	R	None	T_PCIE2_RP_VEND_XP_STATS1_NAKS_RCVD: Counts the number of NAKs received from the endpoint in 64 microseconds. Saturates at 255 (counter does not roll over to 0).
15:8	R	None	T_PCIE2_RP_VEND_XP_STATS1_CRC_ERRORS: Counts the number of CRC errors detected in 64 microseconds. Saturates at 255 (counter does not roll over to 0).
7:0	R	None	T_PCIE2_RP_VEND_XP_STATS1_8B10B_ERRORS: Counts the number of 8b/10b errors detected in one microsecond. Saturates at 255 (counter does not roll over to 0).

34.4.8.54 T_PCIE2_RP_VEND_ERROR_COUNT

This register contains 8-bit saturating counters for some of RXL and TXBA reported errors.

Offset: 0xF60 | Read/Write: R

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R	None	T_PCIE2_RP_VEND_ERROR_COUNT_REPLAY: Counts the number of times that TXBA replays. Saturates at 8'hFF. Can be cleared by writing 1 to PRIV_ERRSTS_REPLAY_STARTED_ERR.
15:8	R	None	T_PCIE2_RP_VEND_ERROR_COUNT_BAD_TLP: Counts bad tlps reported by RXL (sum of lcrcl_err and bad_seq). Saturates at 8'hFF. Can be cleared by writing 1 to ERPTCAP_CERR_BAD_TLP
7:0	R	None	T_PCIE2_RP_VEND_ERROR_COUNT_LCRC_ERR: Counts LCRCL Errors reported by RXL. Saturates at 8'hFF. Can be cleared by writing 1 to PRIV_ERRSTS_LCRC_ERR.

34.4.8.55 T_PCIE2_RP_CFG_MISC

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: 0xF64 | Read/Write: R/W

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:0	R/W	0h	T_PCIE2_RP_CFG_MISC_MUTE_IDLE: MUTE mode is a power management feature in which cpu_clk and m2clk can be gated, if all the units are idle. A particular unit is idle when it has no outstanding transactions to be sent out. The MUTE_IDLE field is used to delay the idle assertion from xvr to clk. It should be programmed to cover the synchronization delay from cpu_clk to m2clk. 0h: MUTE_IDLE_MIN (default) FFh: MUTE_IDLE_MAX

34.4.8.56 T_PCIE2_RP_PRIV_XP_INIT_RECOVERY

Each new time frame, the number of 8B/10B errors is counted. As soon as the number of 8B/10B errors is equal to the threshold, the link is sent into recovery. If at the end of the time frame, the number of 8B/10B errors is smaller than the threshold, a new time frame is started in which the 8B/10B errors are counted.

Offset: 0xF68 | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	T_PCIE2_RP_PRIV_XP_INIT_RECOVERY_8B10B_ERROR_ENABLE: This bit enables the feature T_PCIE2_RP_ERRORRATE_8B10B_ENABLE_OFF disables the feature T_PCIE2_RP_ERRORRATE_8B10B_ENABLE_ON enables the feature 0h: 8B10B_ERROR_ENABLE_INIT (default)
30:20	R/W	64h	T_PCIE2_RP_PRIV_XP_INIT_RECOVERY_8B10B_ERROR_WINDOW: Programs the timeframe in multiples of 1 microsecond. 64h: 8B10B_ERROR_WINDOW_100US (default)
19:0	R/W	9C4h	T_PCIE2_RP_PRIV_XP_INIT_RECOVERY_8B10B_ERROR_THRESHOLD: Bit [21:1] programs the 8B/10B error threshold. If the number of errors in the programmed timeframe is equal to the threshold, the link is sent into recovery. 9C4h: 8B10B_ERROR_THRESHOLD_2500 (default)

34.4.8.57 T_PCIE2_RP_PRIV_XP_LCTRL_2

Offset: 0xF6C | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	None	T_PCIE2_RP_PRIV_XP_LCTRL_2_UPCONFIGURE_CAPABLE: Gen2 link width up-configure capability of remote device as recorded from data rate id of received TS ordered sets.
30	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_REV2P0_COMPLIANCE_DIS: To disable advertising rev2.0 support and link width up-configure capability. 0h: REV2P0_COMPLIANCE_DIS_INIT (default)
29	R/W	1h	T_PCIE2_RP_PRIV_XP_LCTRL_2_EIDLE_INFERENCE_EN: To disable electrical idle inference. It is enabled by default. 1h: EIDLE_INFERENCE_EN_INIT (default)
28	R/W	1h	T_PCIE2_RP_PRIV_XP_LCTRL_2_SURPRISE_IDLE_USE_STAT_IDLE: When set, use idle status signals from the pad to detect surprise electrical idle entry when running in Gen1 speed. 1h: SURPRISE_IDLE_USE_STAT_IDLE_INIT (default)
27	R/W	1h	T_PCIE2_RP_PRIV_XP_LCTRL_2_ALLOW_SPEED_CHANGE_FROM_L1: When set to 1, allow speed change to be initiated from the L1 or Rx_L0s link states. 1h: ALLOW_SPEED_CHANGE_FROM_L1_INIT (default)
26:24	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_RECOVERY_SPEED_TIMEOUT_ADJ: To adjust the minimum length of time the transmitter stays in electrical idle in the Recover.Sped state. The value denotes the time in microseconds added. 0h: RECOVERY_SPEED_TIMEOUT_ADJ_INIT (default)
23:20	R/W	6h	T_PCIE2_RP_PRIV_XP_LCTRL_2_N_EIE_SYMBOLS: This represents the number of K28.7 symbols transmitted prior to transmitting the first FTS ordered set in the Tx.L0s.FTS state when running in Gen2 speed. 6h: N_EIE_SYMBOLS_INIT (default)
19	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_DEEMPHASIS_STRAP: This is the backdoor register for BIOS to write an initial value to the HWInit register PCIE2_RP_LINK_CONTROL_STATUS_2_SELECTABLE_DEEMPHASIS. 0 = -6db 1 = -3.5db 0h: DEEMPHASIS_STRAP_INIT (default)

Bit	R/W	Reset	Description
18	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_ENFORCE_DEEMPHASIS: Forces root port to use de-emphasis value specific in Link Control 2 Selectable De-emphasis field instead of the value requested by the endpoint (advertised in TSs). This bit is reserved for endpoint. Default value is 0b. Functionality of this bit is no longer needed since TX_PEAK_R2_SEL1/SEL0 registers provide complete control over AMP/PEAK/PEAK_PRE settings in gen2. NOTE: this field is reset by Cold Reset 0h: ENFORCE_DEEMPHASIS_INIT (default)
17	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_POLLING_PREDETERMINED_LANES: When set to 1, only lane 0 or 15 needs to detect an exit from electrical idle before moving from Polling.Active to Polling.Config. This bit is not yet implemented. 0h: POLLING_PREDETERMINED_LANES_DISABLE (default)
16	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_AUTONOMOUS_CHANGE: When set to 1, it indicates that the Upstream port-initiated link width/speed change is not caused by a link reliability issue. 0h: AUTONOMOUS_CHANGE_INIT (default)
15:12	R	1h	T_PCIE2_RP_PRIV_XP_LCTRL_2_DATA_RATE_SUPPORTED_REMOTE: Gen2 data rate supported by other side as recorded from data rate identifier of received TS ordered sets. 1h: DATA_RATE_SUPPORTED_REMOTE_2P5 (default) 2h: DATA_RATE_SUPPORTED_REMOTE_5P0_2P5
11:8	R/W	2h	T_PCIE2_RP_PRIV_XP_LCTRL_2_DATA_RATE_SUPPORTED: When set to 4'b01, the supported link speed reported in the Link Capabilities Register is 2.5 Gb/s. When set to 4'b10, the supported link speeds reported in the Link Capabilities Register are 5.0 Gb/s and 2.5 Gb/s. 1h: DATA_RATE_SUPPORTED_2P5 2h: DATA_RATE_SUPPORTED_5P0_2P5 (default)
7:4	R/W	2h	T_PCIE2_RP_PRIV_XP_LCTRL_2_TARGET_LINK_SPEED: Specifies the link rate to change to. This is reflected in the data rate advertised in the data rate identifier field of TS ordered sets. 1h: TARGET_LINK_SPEED_2P5 2h: TARGET_LINK_SPEED_5P0 (default)
3	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_CTL_TX_MARGIN_OVERRIDE: 0h: CTL_TX_MARGIN_OVERRIDE_DISABLED (default) 1h: CTL_TX_MARGIN_OVERRIDE_ENABLED
2	R	0	Reserved
1	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_ADVERTISED_RATE_CHANGE: A write of 1 to this bit triggers the LTSSM to enter Recovery state, without setting the speed change bit, to change the advertised rate. This bit returns a value of 0 on a read. 0h: ADVERTISED_RATE_CHANGE_ZERO (default)
0	R/W	0h	T_PCIE2_RP_PRIV_XP_LCTRL_2_SPEED_CHANGE: A write of 1 to this bit triggers the link speed negotiation procedure. This bit returns a value of 0 on a read. 0h: SPEED_CHANGE_ZERO (default)

34.4.8.58 T_PCIE2_RP_PRIV_XP_PAD_PWRUP

Offset: 0xF74 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	T_PCIE2_RP_PRIV_XP_PAD_PWRUP_PMRX_PWRUP_THRESHOLD: It is the time to hold CDR at reset in Recovery.RcvrLock while waiting for power rails to be stable 0h: PMRX_PWRUP_THRESHOLD_INIT (default)
15:0	R	0	Reserved

34.4.8.59 T_PCIE2_RP_PRIV_XP_RECOVERY_REASONS

This register records the reasons for going into the recovery state. Each bit represents a different reason. Search for "recovery_reason" in the linkfsm to see what each bit means, as it likely will change from project to project. Writing a 0 clears the register and enters into "record ALL recoveries encountered" mode. Writing a 1 clears the register and enters into "record NEXT recovery encountered" mode. The default mode is "record ALL"

Offset: 0xF84 | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	T_PCIE2_RP_PRIV_XP_RECOVERY_REASONS_VALUE: 0h: VALUE_INIT (default) 0h: VALUE_REC_ALL 1h: VALUE_REC_NEXT

34.4.8.60 T_PCIE2_RP_PRIV_XP_RECOVERY_COUNT

This register reflects the number of RECOVERY STATE entries in XP by the XVR. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: 0xF88 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_RECOVERY_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.61 T_PCIE2_RP_PRIV_XP_RX_L0S_ENTRY_COUNT

This register reflects the number of entries from L0 to L0s at LTSSM RX side. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: 0xF8C | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_RX_L0S_ENTRY_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.62 T_PCIE2_RP_PRIV_XP_TX_L0S_ENTRY_COUNT

This register reflects the number of entries from L0 to L0s at LTSSM TX side. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: 0xF90 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_TX_L0S_ENTRY_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.63 T_PCIE2_RP_PRIV_XP_L1_ENTRY_COUNT

This register reflects the number of entries from L0 to L1 It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read

Offset: 0xF94 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_L1_ENTRY_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.64 T_PCIE2_RP_PRIV_XP_L1_TO_RECOVERY_COUNT

This register reflects the number of entries from L1 to Recovery. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: 0xF98 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_L1_TO_RECOVERY_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.65 T_PCIE2_RP_PRIV_XP_L0_TO_RECOVERY_COUNT

This register reflects the number of entries from L0 to Recovery. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: 0xF9C | Read/Write: R/W

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	0h	T_PCIE2_RP_PRIV_XP_L0_TO_RECOVERY_COUNT_REASON: 0h: REASON_INIT (default) 0h: REASON_ALL 1h: REASON_ERR
7:0	R	0h	T_PCIE2_RP_PRIV_XP_L0_TO_RECOVERY_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.66 T_PCIE2_RP_PRIV_XP_L1P_ENTRY_COUNT

This register reflects the number of entries from L0 to Deep L1. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: 0xFA0 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_L1P_ENTRY_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.67 T_PCIE2_RP_PRIV_XP_ASLM_COUNT

This register reflects the number of switching between x1 and x16 for Gen2 Only. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: 0xFA4 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_ASLM_COUNT_VALUE: 0h: VALUE_INIT (default)

34.4.8.68 T_PCIE2_RP_RP_VEND_CTL2

This register contains miscellaneous control fields.

Offset: 0xFA8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:25	R	0h	Reserved

Bit	R/W	Reset	Description
24	R/W	0h	T_PCIE2_RP_VEND_CTL2_COMPLIANCE_X8_DELAY: CTL for compliance delay pattern If set to 0, delay pattern will wrap as soon as the pattern hits lane n - 1 where n is the maximum width of the controller for the current xbar configuration. This ensures that there is always at least one lane sending a delay pattern during compliance. If set to 1, delay pattern will wrap when the pattern hits lane 8 or 16 (mod 8) even if its maximum width is narrower than x8. This means there will be stretches when no lane is sending a delay pattern during compliance. 0h: COMPLIANCE_X8_DELAY_INIT (default)
23	R/W	1h	T_PCIE2_RP_VEND_CTL2_IGNORE_ATTENTION_BUTTON_MSG: When this bit is set, the DUT will ignore upstream Attention button pressed message which has been sent by the endpoint. PCIe specification 1.1 specifies that this message should be ignored. 0h: IGNORE_ATTENTION_BUTTON_MSG_FALSE 1h: IGNORE_ATTENTION_BUTTON_MSG_TRUE (default)
22	R/W	0h	T_PCIE2_RP_VEND_CTL2_UNBLOCK_UP_TRANSACTIONS: When an upstream error has occurred which causes all further upstream transactions to be blocked, writing one to this bit will clean blocking of all upstream transaction. 0h: UNBLOCK_UP_TRANSACTIONS_FALSE (default) 1h: UNBLOCK_UP_TRANSACTIONS_TRUE
21	R/W	0h	T_PCIE2_RP_VEND_CTL2_BLOCK_UP_TRANSACTIONS_ON_ERR: To block all upstream transactions after an Error has been detected and forwarding of packet after the error can cause data corruption or break programming paradigm. 0h: BLOCK_UP_TRANSACTIONS_ON_ERR_EN (default) 1h: BLOCK_UP_TRANSACTIONS_ON_ERR_DIS
20	R/W	0h	T_PCIE2_RP_VEND_CTL2_HW_AUTO_WIDTH_DISABLE_DIS: To hide the LINK_CONTROL_STATUS_HW_AUTO_WIDTH_DISABLE register. 1h: HW_AUTO_WIDTH_DISABLE_DIS_TRUE 0h: HW_AUTO_WIDTH_DISABLE_DIS_FALSE (default)
19	R/W	0h	T_PCIE2_RP_VEND_CTL2_BW_MANAGEMENT_INT_EN_DIS: To hide the LINK_CAPABILITIES_BW_MANAGEMENT_INT_EN register. 1h: BW_MANAGEMENT_INT_EN_DIS_TRUE 0h: BW_MANAGEMENT_INT_EN_DIS_FALSE (default)
18	R/W	0h	T_PCIE2_RP_VEND_CTL2_AUTO_BANDWIDTH_INT_EN_DIS: To hide the LINK_CAPABILITIES_AUTO_BANDWIDTH_INT_EN register. 1h: AUTO_BANDWIDTH_INT_EN_DIS_TRUE 0h: AUTO_BANDWIDTH_INT_EN_DIS_FALSE (default)
17	R/W	0h	T_PCIE2_RP_VEND_CTL2_AUTO_BANDWIDTH_DIS: To hide the LINK_CAPABILITIES_AUTO_BANDWIDTH register. 1h: AUTO_BANDWIDTH_DIS_TRUE 0h: AUTO_BANDWIDTH_DIS_FALSE (default)
16	R/W	0h	T_PCIE2_RP_VEND_CTL2_BW_MANAGEMENT_DIS: To hide the LINK_CAPABILITIES_BW_MANAGEMENT register. 1h: BW_MANAGEMENT_DIS_TRUE 0h: BW_MANAGEMENT_DIS_FALSE (default)
15:14	R	0	Reserved
13	R/W	1h	T_PCIE2_RP_VEND_CTL2_SHADOW_LINK_BW_NOTIFY_CAP: 1h: SHADOW_LINK_BW_NOTIFY_CAP_EN (default) 0h: SHADOW_LINK_BW_NOTIFY_CAP_DIS
12	R	0h	T_PCIE2_RP_VEND_CTL2_GEN2_PROTOCOL_DISABLE: 1h: ON 0h: OFF (default)
11	R/W	1h	T_PCIE2_RP_VEND_CTL2_SLOT_IMPLEMENTED: 1h: SLOT_IMPLEMENTED_INIT (default) 1h: SLOT_IMPLEMENTED_YES 0h: SLOT_IMPLEMENTED_NO
10	R	0	Reserved
9	RW1C	0h	T_PCIE2_RP_VEND_CTL2_ERR_STS_HOTPLUG_PME: NOTE: this field is reset by Cold Reset 0h: ERR_STS_HOTPLUG_PME_FALSE (default) 1h: ERR_STS_HOTPLUG_PME_TRUE 1h: ERR_STS_HOTPLUG_PME_CLEAR

Bit	R/W	Reset	Description
8	RW1C	0h	T_PCIE2_RP_VEND_CTL2_ERR_STS_HOTPLUG_NMI: NOTE: this field is reset by Cold Reset 0h: ERR_STS_HOTPLUG_NMI_FALSE (default) 1h: ERR_STS_HOTPLUG_NMI_TRUE 1h: ERR_STS_HOTPLUG_NMI_CLEAR
7	R/W	0h	T_PCIE2_RP_VEND_CTL2_PCA_ENABLE: This bit will enable/disable the PCA in any TMS. CTLR0's bit controls the enable/disable in a TMS. Default value is 1'b1(ENABLE_ON). 1h: PCA_ENABLE_ON 0h: PCA_ENABLE_OFF (default)
6	R	0	Reserved
5	R/W	0h	T_PCIE2_RP_VEND_CTL2_GEN2_SPEED_DISABLE: This bit will limit the maximum link speed to Gen1 when set. 1h: GEN2_SPEED_DISABLE_ON 0h: GEN2_SPEED_DISABLE_OFF (default)
4	R/W	0h	T_PCIE2_RP_VEND_CTL2_GEN2_PROTOCOL_DISABLE: This will disable Gen2 protocol and will force the use of Gen 1.1 protocol. 0h: GEN2_PROTOCOL_DISABLE_OFF (default) 1h: GEN2_PROTOCOL_DISABLE_ON
3	R	0	Reserved
2	R/W	1h	T_PCIE2_RP_VEND_CTL2_DEV_CAP_EXTENDED_TAG_FIELD_SIZE: This is shadow field of the T_PCIE2_RP_DEVICE_CAPABILITY_EXTENDED_TAG_FIELD_SIZE. Values written to this field will be reflected in EXTENDED_TAG_FIELD_SIZE of T_PCIE2_RP_DEVICE_CAPABILITY. 1h: DEV_CAP_EXTENDED_TAG_FIELD_SIZE_8B (default) 0h: DEV_CAP_EXTENDED_TAG_FIELD_SIZE_5B
1	R/W	0h	T_PCIE2_RP_VEND_CTL2_DIS_MA_TA_EP_MERGE: When set disables the logic which takes care of merging of EP/MA/TA when merging is enabled on CPL packet (downstream). 1h: DIS_MA_TA_EP_MERGE_ON 0h: DIS_MA_TA_EP_MERGE_OFF (default)
0	R	0	Reserved

34.4.8.69 T_PCIE2_RP_PRIV_XP_CONFIG

Offset: 0xFAC | Read/Write: R/W

Bit	R/W	Reset	Description
31:2	R	0	Reserved
1:0	R/W	0h	T_PCIE2_RP_PRIV_XP_CONFIG_LOW_PWR_DURATION: Meant for selection of different types of the information logging for the Health and Performance counters. Primarily, this register supports the selection of different information for the low power duration count. It is used for selecting the Low power information to be presented in the T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS register. TX_L0S, RX_L0S and L1 collect the duration for which LTSSM was in that state. DURATION_IDLE selection gives following IDLE = Duration(TX_L0S && RX_L0S) + Duration of L1 0h: LOW_PWR_DURATION_TX_L0S (default) 1h: LOW_PWR_DURATION_RX_L0S 2h: LOW_PWR_DURATION_L1 3h: LOW_PWR_DURATION_IDLE

34.4.8.70 T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS

This register counts the duration in multiples of 100ns for the type of low power information selected by the T_PCIE2_RP_PRIV_XP_CONFIG_LOW_PWR_DURATION register.

Note: *This actually reports in multiples of clk25m time periods that fit into 100ns:*

Actual time = int(100 / clk25m time period) * 100NS_VALUE

For example, when clk25m = 12 MHz, the time period is 83.33 ns, so this status register reports time = 100_NS_VALUE x 83.33 ns.

Offset: 0xFB0 | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS_VALUE: 0h: VALUE_INIT (default)

34.4.8.71 T_PCIE2_RP_PRIV_XP_SKP_TIMEOUT

This register counts the duration in multiples of 100ns for the type of low power information selected by the T_PCIE2_RP_PRIV_XP_CONFIG_LOW_PWR_DURATION.

Offset: 0xFB4 | Read/Write: R/W

Bit	R/W	Reset	Description
31:26	R	0	Reserved
25:13	R/W	569h	T_PCIE2_RP_PRIV_XP_SKP_TIMEOUT_THRESHOLD_GEN2: 569h: THRESHOLD_GEN2_INIT (default) 271h: THRESHOLD_GEN2_250 2B4h: THRESHOLD_GEN2_278 30Dh: THRESHOLD_GEN2_313 37Ah: THRESHOLD_GEN2_357 410h: THRESHOLD_GEN2_417 4DFh: THRESHOLD_GEN2_500 569h: THRESHOLD_GEN2_555
12:0	R/W	569h	T_PCIE2_RP_PRIV_XP_SKP_TIMEOUT_THRESHOLD: 569h: THRESHOLD_INIT (default) 4E2h: THRESHOLD_250 569h: THRESHOLD_278 61Ah: THRESHOLD_313 6F5h: THRESHOLD_357 821h: THRESHOLD_417 9BFh: THRESHOLD_500 AD2h: THRESHOLD_555

34.4.8.72 T_PCIE2_RP_PRIV_XP_EIDLE_INFERENCE_TIMEOUT

Offset: 0xFB8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	6F2h	T_PCIE2_RP_PRIV_XP_EIDLE_INFERENCE_TIMEOUT_B: This sets the UI window for electrical idle exit not detected in the Recover.Speed state on an unsuccessful speed negotiation. 6F2h: B_INIT (default)
15:0	R/W	8Fh	T_PCIE2_RP_PRIV_XP_EIDLE_INFERENCE_TIMEOUT_A: This sets the UI window for COM not detected in the Recovery.Speed state on a successful speed negotiation. 8Fh: A_INIT (default)

34.4.8.73 T_PCIE2_RP_VEND_SHADOW

This register is the shadow register of T_PCIE2_RP_SS_1. It must be programmed at boot time with the SubSystemVendorID and a unique non-zero number assigned by that vendor for this product. The T_PCIE2_RP_SS_1 register must be read-only at runtime in config space.

Offset: 0xFC8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	0h	T_PCIE2_RP_VEND_SHADOW_SS_1_SSID: 0: SSID_INIT (default)
15:0	R/W	10DEh	T_PCIE2_RP_VEND_SHADOW_SS_1_SSVID: 10DEh: SSVID_INIT (default)

34.4.8.74 T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP

The MAP0 and MAP1 registers contain the mapping of the Transmit Margin field in the Link Control 2 register to the TX_AMP signal which will be driven onto the pads.

Offset: 0xFCC | Read/Write: R/W

Bit	R/W	Reset	Description
31:30	R/W	1h	T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CTL: 1h: CTL_INIT (default)
29:24	R/W	15h	T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE3: 15h: CODE3_INIT (default) 15h: CODE3_DESKTOP 8h: CODE3_MOBILE
23:22	R	0	Reserved
21:16	R/W	18h	T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE2: 18h: CODE2_INIT (default) 18h: CODE2_DESKTOP Ah: CODE2_MOBILE
15:14	R	0	Reserved
13:8	R/W	1Bh	T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE1: 1Bh: CODE1_INIT (default) 1Bh: CODE1_DESKTOP Ch: CODE1_MOBILE
7:6	R	0	Reserved
5:0	R/W	1Fh	T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE0: 1Fh: CODE0_INIT (default) 1Fh: CODE0_DESKTOP Ch: CODE0_MOBILE

34.4.8.75 T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP

Offset: 0xFD0 | Read/Write: R/W

Bit	R/W	Reset	Description
31:30	R/W	1h	T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CTL: 1h: CTL_INIT (default)
29:24	R/W	9h	T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE7: 9h: CODE7_INIT (default) 9h: CODE7_DESKTOP 0h: CODE7_MOBILE
23:22	R	0	Reserved
21:16	R/W	Ch	T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE6: Ch: CODE6_INIT (default) Ch: CODE6_DESKTOP 2h: CODE6_MOBILE
15:14	R	0	Reserved
13:8	R/W	Fh	T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE5: Fh: CODE5_INIT (default) Fh: CODE5_DESKTOP 4h: CODE5_MOBILE
7:6	R	0	Reserved
5:0	R/W	11h	T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE4: 11h: CODE4_INIT (default) 11h: CODE4_DESKTOP 6h: CODE4_MOBILE

34.4.8.76 T_PCIE2_RP_TIMEOUT2

Offset: 0xFDC | Read/Write: R/W

Bit	R/W	Reset	Description
31:5	R	0	Reserved
4:0	R/W	Ah	T_PCIE2_RP_TIMEOUT2_MIN_L1_L2_IDLE_TIME: Sets the minimum amount of time to stay in L1/L2. Holds off exit-condition detection for this amount of time to prevent accidental wake from data before E-Idle (second EIOS in Gen2) Measured in units of 4ns scaled with xclk frequency. Ah: MIN_L1_L2_IDLE_TIME_INIT (default)

34.4.8.77 T_PCIE2_RP_PRIV_MISC

Offset: 0xFE0 | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	T_PCIE2_RP_PRIV_MISC_TMS_CLK_CLAMP_ENABLE: Enables per-TMS XCLK/XTXCLK clamping using value from T_PCIE2_RP_PRIV_MISC_CLK_CLAMP_THRESHOLD to determine when to clamp. This will clamp all the logic inside a TMS shared between controllers. Note: This enable is per TMS. Use this TMS's controller 0's register to set this. 0h: TMS_CLK_CLAMP_ENABLE_INIT (default)
30:24	R/W	3h	T_PCIE2_RP_PRIV_MISC_TMS_CLK_CLAMP_THRESHOLD: Number of contiguous nbref_clks (25MHz) after clamping requirements are met (L1, lanes powered down if necessary, TX FIFO empty, no eidle exit, etc.) before actually clamping xclk/xtxclk. Choose a value large enough to allow all upstream transactions to clear the XP (make it into the FPCI clock domain) and for the sleep signals coming from the LTSSM to make it to the pads. If the value is too small, upstream transactions will get stuck in the XP and/or the sleep signals will not assert properly. The value must also be large enough so that config transactions have time to propagate from the pri domain to xclk domain. Note: This enable is per TMS. Use this TMS's controller 0's register to set this. 3h: TMS_CLK_CLAMP_THRESHOLD_INIT (default)
23	R/W	0h	T_PCIE2_RP_PRIV_MISC_CTLR_CLK_CLAMP_ENABLE: Enables per-controller XCLK/XTXCLK clamping using value from T_PCIE2_RP_PRIV_MISC_CTLR CLK_CLAMP_THRESHOLD to determine when to clamp. 0h: CTLR_CLK_CLAMP_ENABLE_INIT (default)
22:16	R/W	1h	T_PCIE2_RP_PRIV_MISC_CTLR_CLK_CLAMP_THRESHOLD: Number of contiguous nbref_clks (12MHz) after clamping requirements are met (L1, lanes powered down if necessary, TX FIFO empty, no eidle exit, etc.) before actually clamping xclk/xtxclk. Choose a value large enough to for the sleep signals coming from the LTSSM to make it to the pads. If the value is too small, the sleep signals will not assert properly. 1h: CTLR_CLK_CLAMP_THRESHOLD_INIT (default)
15:4	R	0	Reserved
3:0	R/W	Fh	T_PCIE2_RP_PRIV_MISC_PRSNT_MAP. Drives the present status to the AFI: 1101 = No component present (Controller's internal post_x_bar_prsnt_ signal is hardwired to 1) 1110 = Component is present (Controller's internal post_x_bar_prsnt_ signal is hardwired to 0) 1111 = Default - post_x_bar_prsnt_ is no component present. (Controller's internal post_x_bar_prsnt_ signal is hardwired to 1) Fh: PRSNT_MAP_INIT (default)

34.4.8.78 T_PCIE2_RP_VEND_XP3

This register is the Symbol Alignment Error Handling register.

Offset: 0xFE8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:8	R	0	Reserved

Bit	R/W	Reset	Description
7:0	R/W	3h	T_PCIE2_RP_VEND_XP3_SA_ERROR_LIMIT: Specifies the number of misaligned COM symbols needed before declaring loss of Symbol Alignment. When Symbol Alignment is lost, the error bit for all subsequent symbols is asserted until 1 good COM symbol is seen. Then the error bit is deasserted and the counter tracking misaligned COM symbols is reset. Set to 0 to disable this feature. Note: Since 8B/10B and symbol alignment errors are indistinguishable, symbol alignment errors (when error limit is hit) will contribute to all 8B/10B error counts including the Goto Recovery on 8B/10B errors feature. This is useful to speed up link retraining after losing symbol alignment. 3h: SA_ERROR_LIMIT_INIT (default)

34.4.8.79 T_PCIE2_RP_XP_CTL_1

This register contains control registers used in XP.

Offset: 0xFEC | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	1h	T_PCIE2_RP_XP_CTL_1_CYA_RP_INITIATED_L2_WAKE_ON_TLP_PENDING: 1h: CYA_RP_INITIATED_L2_WAKE_ON_TLP_PENDING_INIT (default)
30:28	R/W	1h	T_PCIE2_RP_XP_CTL_1_SPARE: 1h SPARE_INIT (default)
27	R/W	1h	T_PCIE2_RP_XP_CTL_1_EXIT_L1_AT_CLKREQ_ASSERTION: When set, the LTSSM will exit L1 when clkreq assertion happens. NOTE : This feature only applies to downstream port 0h: EXIT_L1_AT_CLKREQ_ASSERTION_DISABLED 1h: EXIT_L1_AT_CLKREQ_ASSERTION_ENABLED (default)
26	R/W	1h	T_PCIE2_RP_XP_CTL_1_NEW_IOBIST_CTRL: This bit when set will enable control signals from iobist to take effect. Some of the control signal will be are enabling/disabling scrambling, enabling and disabling checking for SKP symbols. 0h: NEW_IOBIST_CTRL_DISABLED 1h: NEW_IOBIST_CTRL_ENABLED (default)
25	R/W	0h	T_PCIE2_RP_XP_CTL_1_OLD_IOBIST_EN: This bit when set will enable old iobist, else it will enable new iobist. 0h: OLD_IOBIST_EN_INIT (default)
24:19	R/W	0h	T_PCIE2_RP_XP_CTL_1_EIOS_DEBOUNCE_TIMER: This is the number of xclks for which the rx_data_en signals need to be deasserted before asserting again. 0h: EIOS_DEBOUNCE_TIMER_INIT (default)
18	R/W	0h	T_PCIE2_RP_XP_CTL_1_DISABLE_RX_LANE_AFTER_EIOS: Disable rx_lane_en after EIOS is seen in any of the enabled lane during recovery.Rlock and Recovery.Config in order to avoid coupling. 0h: DISABLE_RX_LANE_AFTER_EIOS_INIT (default)
17	R/W	0h	T_PCIE2_RP_XP_CTL_1_ENABLE_DESKEW_FOR_SPDCH_NEGOTIATION: When set to 0, forces LTSSM to disable deskew in Recovery.RcvrCfg state if the transition from Recovery.RcvrLock to RcvrCfg state is due to LTSSM detecting eight consecutive training set on any of the lanes but not all of the lanes, and speed change bit is set to 1. When set to 1, LTSSM will always enable deskew in Recovery.RcvrCfg state. 0h: ENABLE_DESKEW_FOR_SPDCH_NEGOTIATION_INIT (default)
16	R/W	0h	T_PCIE2_RP_XP_CTL_1_BYPASS_CONFIG_PWRUP: When set to 0, forces LTSSM to go from Recovery.RcvrLock to Config.Pwrup if LTSSM sees training set on some but not all the active lanes. This bit only takes effect when the link is operating in dynamically reduced link width and specification-defined link width resize is enabled. When set to 1, LTSSM will go from Recovery.RcvrLock to Config.LinkStart directly, bypassing Config.Pwrup, when LTSSM sees training set on some but not all the active lanes. 0h: BYPASS_CONFIG_PWRUP_INIT (default)
15	R/W	0h	T_PCIE2_RP_XP_CTL_1_RESET_TS_CTRL_ON_ERROR: When set to 1, in Config.LinkStart state, LTSSM will reset the internal TS_CTRL0 register if an 8B/10B error is present on any symbol of the training set. When set to 0, in Config.LinkStart state, LTSSM will reset the internal TS_CTRL0 register if 8B/10B error is present on symbols 0, 4 or 5 only. 0h: RESET_TS_CTRL_ON_ERROR_INIT (default)
14	R/W	0h	T_PCIE2_RP_XP_CTL_1_RX_EIDLE_EXIT_TIMER_IN_CONFIG: When set to 1, T_PCIE2_RP_XP_IDLE_TIMER_1_RX_EIDLE_EXIT will be applied in Config.LinkStart only. When set to 0, T_PCIE2_RP_XP_IDLE_TIMER_1_RX_EIDLE_EXIT will be applied in Config.LinkStart, Recovery and Polling. 0h: RX_EIDLE_EXIT_TIMER_IN_CONFIG_INIT (default)

Bit	R/W	Reset	Description
13	R/W	1h	T_PCIE2_RP_XP_CTL_1_FORCE_RESET_IN_CONFIG: When set to 1, LTSSM will force receiver logic to be reset when enabling lanes in Config.LinkStart state for PCIe 2.0 specification-defined link width upconfiguration. 1h: FORCE_RESET_IN_CONFIG_INIT (default)
12	R/W	1h	T_PCIE2_RP_XP_CTL_1_LINK_RESIZE_PWRDN_CTL: 1h: LINK_RESIZE_PWRDN_CTL_INIT (default)
11	R/W	1h	T_PCIE2_RP_XP_CTL_1_ALLOW_SPEED_CHANGE_FROM_L0S: When set to 1, LTSSM will initiate link speed/width/advertised rate change while RX side is in L0s link state. 1h: ALLOW_SPEED_CHANGE_FROM_L0S_INIT (default)
10	R/W	0h	T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_DL_RETRAIN: When set to 1, LTSSM will wait for framer to be idle before entering Recovery for DL initiated link retrain. Note: Setting this bit to 1 could prevent LTSSM to enter Recovery on a replay rollover event if the retry buffer is very full. 0h: PRESERVE_TX_PACKET_ON_DL_RETRAIN_INIT (default)
9	R/W	0h	T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_RECOVERY: Note: Setting this bit to 1 does not work if L1 ASPM is enabled as it will cause hang during L1 negotiation if Root Port initiate entry to Recovery. Note: During DL init sequence, if link is retrained, setting this bit to 1 would hang the chip. 0h: PRESERVE_TX_PACKET_ON_RECOVERY_INIT (default)
8	R/W	1h	T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_WIDTH_CHANGE: When set to 1, LTSSM will wait for framer to be idle before entering Recovery for link width change. 1h: PRESERVE_TX_PACKET_ON_WIDTH_CHANGE_INIT (default)
7	R/W	1h	T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_SPEED_CHANGE: When set to 1, LTSSM will wait for framer to be idle before entering Recovery for link speed change or advertised rate change. 1h: PRESERVE_TX_PACKET_ON_SPEED_CHANGE_INIT (default)
6	R/W	0h	T_PCIE2_RP_XP_CTL_1_RESET_LANE_ENABLE_ORIG_IN_DETECT: This field will allow lane_enable_original to be reset in Detect state. 0h: RESET_LANE_ENABLE_ORIG_IN_DETECT_INIT (default)
5:2	R/W	0h	T_PCIE2_RP_XP_CTL_1_IDLE_TO_L0_DELAY: This field sets the number of clocks that the LTSSM will delay the transition from Recovery.Idle to L0.Normal link state or Config.Idle to L0.Normal link state. 0h: IDLE_TO_L0_DELAY_INIT (default)
1	R/W	0h	T_PCIE2_RP_XP_CTL_1_LWLO_HUNT_ON_BAD_TS1: When set to 1, the LWLOFSM hunt timer does not reset when any lane receives an error on the TS1 ordered set in the Config.LinkStart, Config.LinkAccept, Config.LaneWait, and Config.LaneAccept states. When set to 0, the LWLOFSM hunt timer will reset when any lane receives the first error on the TS1 ordered set after any lanes has locked and hunt timer has started counting in the Config.LinkStart, Config.LinkAccept, Config.LaneWait, and Config.LaneAccept states. 0h: LWLO_HUNT_ON_BAD_TS1_INIT (default)
0	R/W	0h	T_PCIE2_RP_XP_CTL_1_FORCE_SA_IN_CONFIG: When set to 1, enables symbol alignment in Config.LinkAccept, Config.LaneWait, Config.LaneAccept, and Config.Complete states. 0h: FORCE_SA_IN_CONFIG_INIT (default)

34.4.8.80 T_PCIE2_RP_PRIV_XP_L1BEACON

Offset: 0xFF0 | Read/Write: R/W

Bit	R/W	Reset	Description
31:10	R	0	Reserved
9:2	R/W	0h	T_PCIE2_RP_PRIV_XP_L1BEACON_N_EIE_SYMBOLS: It is the number of EIE symbols sent in L1 Beacon Entry/Exist state 40h: N_EIE_SYMBOLS_INIT (default)
1	R/W	0h	T_PCIE2_RP_PRIV_XP_L1BEACON_TX_BYP_CTRL: 0h: TX_BYP_CTRL_DEFAULT (default) 0h: TX_BYP_CTRL_ONLYL0 1h: TX_BYP_CTRL_ALLLANES
0	R/W	0h	T_PCIE2_RP_PRIV_XP_L1BEACON_EN: 0h: EN_DEFAULT (default)

34.4.8.81 T_PCIE2_RP_TIMEOUT3

Offset: 0xFF4 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	Ch	T_PCIE2_RP_TIMEOUT3_RX_L0S_IDLE_TIME_GEN2: This field controls the amount of time LTSSM delays the detection of electrical idle exit in the RX_L0s state at Gen2 speed of operation. Ch: RX_L0S_IDLE_TIME_GEN2_INIT (default)
23:16	R/W	4h	T_PCIE2_RP_TIMEOUT3_RX_L0S_IDLE_TIME_GEN1: This field controls the amount of time LTSSM delays the detection of electrical idle exit in the RX_L0s state at Gen1 speed of operation. 4h: RX_L0S_IDLE_TIME_GEN1_INIT (default)
15:8	R/W	FFh	T_PCIE2_RP_TIMEOUT3_TX_L0S_IDLE_TIME: This field controls the minimum amount of time LTSSM stays in TX_L0S_IDLE state. Measured in units of clk25m clock periods (83 ns when using 12 MHz clock) scaled with xclk frequency. A setting of 8'hFF disables the delay. FFh: TX_L0S_IDLE_TIME_INIT (default)
7:4	R/W	0h	T_PCIE2_RP_TIMEOUT3_RX_EIDLE_EXIT_DELAY: This field controls the amount of time the LTSSM delays the detection of TS ordered set when exiting out of electrical idle state. Each increment represents 2^X microseconds. A setting of 0 disables the delay. 0h: RX_EIDLE_EXIT_DELAY_INIT (default)
3:0	R/W	0h	T_PCIE2_RP_TIMEOUT3_TX_EIDLE_EXIT_DELAY: This field controls the amount of time the LTSSM delays the transmission of TS ordered sets when exiting out of electrical idle state. Unit is in microseconds. Each increment represents 2^X microseconds. A setting of 0 disables the delay. 0h: TX_EIDLE_EXIT_DELAY_INIT (default)

34.4.8.82 T_PCIE2_RP_XP_CTL_2

This register contains controls bits used in XP.

Offset: 0xFF8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23	R/W	1h	T_PCIE2_RP_XP_CYA_2_RX_CAL_EN 1h: RX_CAL_EN_INIT
22	R/W	1h	T_PCIE2_RP_XP_CYA_2_AUX_RX_IDLE_EN 1h: RX_IDLE_EN_INIT
21	R/W	0h	T_PCIE2_RP_XP_CYA_2_OVERRIDE_AUX_RX_IDLE_EN 0h: RX_IDLE_EN_INIT
20	R/W	1h	T_PCIE2_RP_XP_CTL_2_WAIT_FOR_LOCKDET_DURING_L1_EXIT
19:12	R/W	FFh	T_PCIE2_RP_XP_CTL_2_ECO351987_HOLDOFF_CNTR_LIMIT FFh: ECO351987_HOLDOFF_CNTR_LIMIT_INIT (default)
11	R/W	1h	T_PCIE2_RP_XP_CTL_2_USE_QUALIFIED_TS_CTL_BITS 1h USE_QUALIFIED_TS_CTL_BITS_INIT (default)
10:9	R	0	Reserved
8	R/W	0h	T_PCIE2_RP_XP_CTL_2_CONFIG_LINKSTART_REQUIRE_PAD_LANE_NUM. Forces the LTSSM training set detector to require PAD in the lane number symbol of training set in Config.LinkStart state: 0h: INIT (default)
7:0	R/W	40h	T_PCIE2_RP_XP_CTL_2_LOOPBACK_EXIT_THRESHOLD. Programs the maximum amount of time spent in the Loopback.Exit state as loopback slave. Units are in xclk. The setting of 0 disables the timer in Loopback.Exit state such that loopback slave relies on detection of electrical idle data to exit from Loopback. The default is 64 xclks: 40h: INIT (default)

34.4.8.83 T_PCIE2_RP_VEND_XP_STATS2

This register contains fields unique to NVIDIA’s implementation of PCI Express devices.

Offset: 0xFFC | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:0	R/W	0h	T_PCIE2_RP_VEND_XP_STATS2_8B10B_ERR_LANEMASK: Each individual bit represents the corresponding logical lane. When set to 1, the 8B/10B error from that specific lane will be masked out from the counter in T_PCIE2_RP_VEND_XP_STATS1_8B10B_ERRORS 0h: 8B10B_ERR_LANEMASK_INIT (default) FFFEh: 8B10B_ERR_LANEMASK_LN0 FFFDh: 8B10B_ERR_LANEMASK_LN1 FFFBh: 8B10B_ERR_LANEMASK_LN2 FFF7h: 8B10B_ERR_LANEMASK_LN3 FFEFh: 8B10B_ERR_LANEMASK_LN4 FFDFh: 8B10B_ERR_LANEMASK_LN5 FFBFh: 8B10B_ERR_LANEMASK_LN6 FF7Fh: 8B10B_ERR_LANEMASK_LN7

34.5 AFI Registers

34.5.1 AFI_AXI_BARi_SZ_0

There are 9 AXI BAR SIZE registers, where i = 0 through 8.

BAR0: Offset: 0x0 | Read/Write: R/W | Reset: 0x0003f000 (0bxxxxxxxxxxxx0011111000000000000)
BAR1 – BAR5: Offset: 0x4 + (i – 1) * 0x4, where i = 1 – 5 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)
BAR6 – BAR8: Offset: 0x134 + (i – 6) * 0x4, where i = 6 – 8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Description
19:0	AXI_BARi_SIZE: The size of the address range associated with BARi is in 4K increments. A value of 0 signifies BARi is not used.

34.5.2 AFI_AXI_BARi_START_0

There are 9 AXI BAR START registers, where i = 0 through 8.

BAR0: Offset: 0x18 | Read/Write: R/W | Reset: 0x01000000 (0b00000001000000000000xxxxxxxxxxxx)
BAR1 – BAR5: Offset: 0x1c + (i – 1) * 0x4, where i = 1 – 5 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)
BAR6 – BAR8: Offset: 0x140 + (i – 6) * 0x4, where i = 6 – 8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Description
31:12	AXI_BARi_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

34.5.3 AFI_FPCI_BARi_0

There are 9 FPCI BAR registers, where i = 0 through 8.

BAR0: Offset: 0x30 | Read/Write: R/W | Reset: 0x00010001 (0b00000000000000010000000000000xxx1)
BAR1 – BAR5: Offset: 0x34 + (i – 1) * 0x4, where i = 1 – 5 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)
BAR6 – BAR8: Offset: 0x14c + (i – 6) * 0x4, where i = 6 – 8 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Description
31:4	FPCI_BARi_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a shift left 8 of the value of this register.

Bit	Description
0	FPCI_BARi_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 1=Memory mapped access (PW only) 0=IO/config access (NPW only)

34.5.4 AFI_MSI_BAR_SZ_0

Do not use this reserved space.

MSI BAR Size

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE

34.5.5 AFI_MSI_FPCI_BAR_ST_0

MSI FPCI BAR Start

Offset: 0x64 | Read/Write: R/W | Reset: 0x58540000 (0b01011000010101000000000000xxxx)

Bit	Reset	Description
31:4	0x5854000	MSI_FPCI_BAR_START: The start of upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

34.5.6 AFI_MSI_AXI_BAR_ST_0

MSI AXI BAR Start

Offset: 0x68 | Read/Write: R/W | Reset: 0x80000000 (0b10000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x80000	MSI_AXI_BAR_START: The start of upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

34.5.7 AFI_MSI_VECi_0

MSI Vector

There are 8 MSI Vector registers (AFI_MSI_VECi), where i is a value from 0 to 7.

Offset: 0x6c + (i * 4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTORi: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0.Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

34.5.8 AFI_MSI_EN_VEC0_0

MSI Enable Vector

There are 8 MSI Enable Vector registers (AFI_MSI_EN_VECi), where i is a value from 0 to 7.

Offset: 0x8c + (i * 4) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTORi: Each vector register corresponds to the enable bit for 32of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

34.5.9 AFI_CONFIGURATION_0

Configuration

Offset: 0xac | Read/Write: R/W | Reset: 0x00fXXX40 (0b0xxxxxxx11111xxx10xxxxxx01000000)

Bit	R/W	Reset	Description
31	RW	0x0	CLKEN_OVERRIDE: This can override the clock enable in case of malfunction.
23	RW	0x1	SPARSE_WSTRB: Diagnostic bit for enabling sparse WSTRB.
22	RW	0x1	UNALIGNED_BYTE_ACCESS: Diagnostic bit for unaligned byte access.
21	RW	0x1	UPSTREAM_RAW_RESPAW_MSIAW: Diagnostic bit for upstream RAW, RESPAW, and MSIAW.
20	RW	0x1	MSG_PLL_PWR_DWN_CYA:
19	RW	0x1	PW_NO_DEVSEL_ERR_CYA: Setting this bit disables detection of DECERR due to no devsel for DS PWs only.
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on AFI upstream. A value of 1b indicates there are no outstanding reads to the initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on AFI upstream. A value of 1b indicates there are no outstanding writes to the initiator.
15	RW	0x1	WDATA_LEAD_CYA: This bit is used to enable/disable the handling of write data ahead of requests on mselect
14	RW	0x0	WR_INTRLV_CYA: This bit is used to enable/disable the handling of interleaved write requests on mselect
13	RO	X	PE1_PRSNT_L_IN: This read-only bit is 0 when a card is present in PCIe slot 1
12	RO	X	PE0_PRSNT_L_IN: This read-only bit is 0 when a card is present in PCIe slot 0
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to AFI target. A value of 1b indicates there are no outstanding reads to the downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to AFI target. A value of 1b indicates there are no outstanding writes to the downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any valid active bits.
7	RW	0x0	UFPCI_MSIAW: MSI After Write ordering rule. 1 = whenever MSI is ready assert the interrupt 0 = default behavior, apply MSIAW ordering rule
6	RW	0x1	UFPCI_PWPASSPW: Input to upstream FPCI 1 = whenever write is ready, send it; 0 = write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to upstream FPCI. Allow upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for upstream FPCI. Allow upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: Used for downstream FPCI. Allow downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: Input to downstream FPCI. Allow downstream FPCI responses to pass writes
1	RW	0x0	DFPCI_PASSPW: Input to downstream FPCI. Allow downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the PCI device block is disabled, it is completely invisible on the PCI bus, i.e., it does not even process PCI configuration accesses.

34.5.10 AFI_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows an FPCI error to be forwarded to AXI response when the FPCI error response indicates Master Abort. 1 = forward error 0 = return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows an FPCI error to be forwarded to AXI response when the FPCI error response indicates Data Error. 1 = forward error 0 = return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows an FPCI error to be forwarded to AXI response when FPCI error response indicates Target Abort. This bit also covers a decode error generated when there is no devsel received. 1 = forward error, 0 = return AXI OKAY response (2'b0)

34.5.11 AFI_INTR_MASK_0

Interrupt Masks

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	MSI_MASK: MSI to CPU complex gated by mask.
0	0x0	INT_MASK: Interrupt to CPU complex gated by mask.

34.5.12 AFI_INTR_CODE_0

Interrupt Code

Detected errors are logged in an error signature register as specified here. The error signature register always stores the first error that occurred after enabling the error capture. The interrupt code register logs the interrupt ID shown in the ID column of Table 217 below. This register stores the ID for the associated interrupt when current ID in the register is zero. Software must write zero to the register to re-enable it to store subsequent interrupts. For ID=6, the capture information is the sideband message bits. There is an associated interrupt generated to the CPU. Since FPCI address is 40 bits, a separate capture register is used for the upper 8 bits of the address.

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: If the code is 0, logging of the next interrupt is enabled. See Table 217 for the system messages. 0 = INT_CODE_CLEAR: Clear interrupt code 1 = INT_CODE_INI_SLVERR: Interrupt code for CPU complex AXI SLVERR response to AFI 2 = INT_CODE_INI_DECERR: Interrupt code for CPU complex AXI DECERR response to AFI 3 = INT_CODE_TGT_SLVERR: Interrupt code for PCIe endpoint FPCI target abort or data error response to AFI 4 = INT_CODE_TGT_DECERR: Interrupt code for PCIe2 FPCI master abort response to AFI 5 = INT_CODE_TGT_WRERR: Interrupt code for bufferable write to non-posted write address region 6 = INT_CODE_SM_MSG: Interrupt code for PCIe2 system management message 7 = INT_CODE_DFPCI_DECERR: Interrupt code for PCIe2 response to downstream request when the downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR: Interrupt code for AFI response to downstream request when the mselect AXI address does not fall in any of AFI downstream BARs 9 = INT_CODE_FPCI_TIMEOUT: Interrupt code for FPCI Timeout 10 = INT_CODE_PE_PRSNT_SENSE: Interrupt code for Slot Present Pin Change 11 = INT_CODE_PE_CLKREQ_SENSE: Interrupt code for Slot Clock Request Change 12 = INT_CODE_CLKCLAMP_SENSE: Interrupt code for TMS Clock Clamp Change 13 = INT_CODE_RDY4PD_SENSE: Interrupt code for TMS Ready for power down 14 = INT_CODE_P2P_ERROR: Interrupt code for Peer2Peer errors

Table 217: System Messages

Error/Message	ID	Capture Information
AXI Slave error	1	Bits [31:2] of Initiator AXI address, direction of transaction
AXI Decode error	2	
AXI Slave error – PCIe target abort or data error	3	Bits [39:2] of Target FPCI address, direction of transaction
AXI Decode error – PCIe master abort	4	
AXI Decode error – write to NPW address region > 4B	5	Bits[39:2] of Target AXI address, direction of transaction
PCIe 2.0 Sideband message	6	Sideband UnitID, Bits[4:0] of sideband message
FPCI Decode error – not in PCIe 2.0 memory space	7	Bits[39:2] of Target FPCI address, direction of transaction
AXI Decode error – not in AFI BAR or register space	8	Bits [31:2] of Target AXI address, direction of transaction
FPCI Timeout	9	None
Slot Present Pin Change	10	Bits [2:0] capture status of slots 2-0 present pin
Slot Clock Request Change	11	Bits [2:0] capture status of slots 2-0 clock request
TMS Clock Clamp Change	12	Bit [0] captures level of clock clamp
TMS Ready for power down	13	None – all 3 controllers are ready for power down
Peer-to-peer error	14	Error response from endpoint device (tms02ufa_cmd_error), Bits[39:2] of Target FPCI address, direction of transaction

34.5.13 AFI_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, this field contains address bits [31:2], either in FPCI memory space or AXI space. If the interrupt code is 6, the information field INT_INFO[12:0] contains sideband information {sideband unitid, 3'b0, tms02sm_msg[4:0]}. For FPCI generated errors, the information field contains FPCI address. For AXI/AFI generated errors, the information field contains AXI address. For interrupt code 10, this field contains the status of slots 2-0 present pin. For interrupt code 11, this field contains the status of slots 2-0 clock request For interrupt code 12, this field contains level of clock clamp
0	0x0	DIR: Indicates the direction of the AXI/FPCI transaction. 1=rd 0=wr If signature type is 6 (sideband message), this field is 1. 0 = WRITE: Interrupt due to a write transaction 1 = READ: Interrupt due to a read transaction

34.5.14 AFI_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00xxxxxxxx00000000)

Bit	Reset	Description
17:16	0x0	P2P_ERR_RESP: These 2 bits are for the captured endpoint device error response (tms02ufa_cmd_error) when the interrupt code is 14.
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of captured FPCI address (bits[39:32]) when the interrupt code is 3, 4, or 7. These bits determine the region in the Hypertransport Address Map that was accessed.

34.5.15 AFI_SM_INTR_ENABLE_0

Sideband Message Interrupt Enable

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000000000000000)

Bit	Reset	Description
14:0	0x0	ENABLE_MESSAGE: Each bit in this register enables a corresponding message shown in the table below.

Table 218: Message Types

[1:0]	[3:2]	[4]	Enable bit	MSG_TYPE	Description
00	00	1	0	Interrupt	INTA Assertion.
	01		1		INTB Assertion.
	10		2		INTC Assertion.
	11		3		INTD Assertion.
	00	0	4		INTA Deassertion
	01		5		INTB Deassertion
	10		6		INTC Deassertion
	11		7		INTD Deassertion
01	00	0	8	Error	Correctable Error
	01		9		Uncorrectable Non-Fatal Error
	10		10		Uncorrectable Fatal Error
10	00	0	11	PME	PME assertion.
10	01	0	12	Hotplug	Hotplug SCI assertion
11	00	1	13	Interrupt	Root Port Interrupt Assertion
11	00	0	14		Root Port Interrupt Deassertion

34.5.16 AFI_AFI_INTR_ENABLE_0

AFI Interrupt Enable

See the AFI_INTR_CODE_0 register for details on the interrupt codes.

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12	0x0	EN_P2P_ERR: Enable bit for interrupt code 14
11	0x0	EN_RDY4PD_SENSE: Enable bit for interrupt code 13
10	0x0	EN_CLKCLAMP_SENSE: Enable bit for interrupt code 12
9	0x0	EN_PE_CLKREQ_SENSE: Enable bit for interrupt code 11
8	0x0	EN_PE_PRSNT_SENSE: Enable bit for interrupt code 10
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

34.5.17 AFI_FPCI_TIMEOUT_0

Do not use this reserved space.

FPCI Timeout

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
31	0x0	SM2TMS0_FPCI_TIMEOUT_EN
19:0	0x0	SM2ALL_FPCI_TIMEOUT_THRESH

34.5.18 AFI_PCIE_THROTTLE_0

PCIe Throttle

Offset: 0xec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx000000000000x000)

Bit	Reset	Description
31	0x0	SM2PCIE_THROT_EN: Override THERM MGMT
15:4	0x0	SM2PCIE_THROT_PERIOD: Override THERM MGMT period
2:0	0x0	SM2PCIE_THROT_DUTY_CYCLE: Override THERM MGMT duty cycle

34.5.19 AFI_PME_0

PCIe PME

Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000XX0 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	R/W	Reset	Description
11	RO	X	TMS0C12SM_PRESENCE_STATE: PCIe Link Presence State
10	RO	X	TMS0C12SM_PME_ACK: PCIe Endpoint PME Ack
9	RO	X	TMS0C12SM_PME: PCIe Endpoint PME message
8	RW	0x0	SM2TMS0C1_PME_TO: SM (system management) to PCIe PME Turn Off
6	RO	X	TMS0C02SM_PRESENCE_STATE: PCIe Link Presence State
5	RO	X	TMS0C02SM_PME_ACK: PCIe Endpoint PME Ack
4	RO	X	TMS0C02SM_PME: PCIe Endpoint PME message
0	RW	0x0	SM2TMS0C0_PME_TO: SM (system management) to PCIe PME Turn Off

34.5.20 AFI_REQ_PENDING_0

Request Pending

Offset: 0xf4 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	TMS0C12SM_NONISO_PENDING: SM (system management) status that a Non-ISO request is pending from PCIe to FPCI
6	X	TMS0C12SM_ISO_PENDING: SM (system management) status that an ISO request is pending from PCIE to FPCI
5	X	TMS0C12SM_NONCOH_REQUEST_PEND: SM (system management) status that a Non-coherent request is pending from PCIe to FPCI
4	X	TMS0C12SM_COH_REQUEST_PEND: SM (system management) status that a coherent request is pending from PCIe to FPCI
3	X	TMS0C02SM_NONISO_PENDING: SM (system management) status that a Non-ISO request is pending from PCIe to FPCI
2	X	TMS0C02SM_ISO_PENDING: SM (system management) status that an ISO request is pending from PCIe to FPCI
1	X	TMS0C02SM_NONCOH_REQUEST_PEND: SM (system management) status that a Non-coherent request is pending from PCIe to FPCI
0	X	TMS0C02SM_COH_REQUEST_PEND: SM (system management) status that a coherent request is pending from PCIe to FPCI

34.5.21 AFI_PCIE_CONFIG_0

PCIe Config

Offset: 0xf8 | Read/Write: R/W | Reset: 0x00103025 (0bxxxxxxxx0001xxx00011xxx00010x101)

Bit	Reset	Description
23:20	0x1	SM2TMS0_XBAR_CONFIG: SM (system management) configuration of PCIe crossbar. 1 = XBAR_4_1: x4_x1 configuration 0 = XBAR_2_1: x2_x1 configuration
16:12	0x3	UNITID_T0C1: T0C1 Upstream FPCI Unit ID. HyperTransport, upstream FPCI request. The downstream FPCI unit ID should remain 0.
8:4	0x2	UNITID_T0C0: T0C0 Upstream FPCI Unit ID. HyperTransport, upstream FPCI request
2	0x1	PCIEC1_DISABLE_DEVICE: Disables PCIe Controller 1 (default on)
1	0x0	PCIEC0_DISABLE_DEVICE: Disables PCIe Controller 0 (default off)
0	0x1	SM2TGIO_SLOT_EMPTY_PD_CYA: Indicates PCIe slot empty. Overrides PCIe slot present input.

34.5.22 AFI_REV_ID_0

Revision ID

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CFG_REVID_WRITE_ENABLE: Write Enable for PCI backdoor rev ID override value.
0	0x0	CFG_REVID_OVERRIDE: Override for PCI config revision ID read-only register. This allows backdoor changes to rev ID for metal spins.

34.5.23 AFI_TOM_0

Top of Memory Limit

Offset: 0x100 | Read/Write: R/W | Reset: 0x3f3f0fff (0bxx11111100111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3f3f	DLDT2ALL_TOM2: Top of Memory Limit 2. Determines peer-to-peer range as:{TOM2: 26'b0} to 0xFC_FFFF_FFFF
11:0	0xfff	DLDT2ALL_TOM1: Top of Memory Limit 1. Determines peer-to-peer range as:{TOM1:: 26'b0} to 0xFFFF_FFFF (except MSI region)

34.5.24 AFI_FUSE_0

PCIe Fuse

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000336 (0bxxxxxxxxxxxxxxxxxxxxxx011x011x11x)

Bit	Reset	Description
10:8	0x3	FUSE_PCIE_WIDTH_T0C1: Configure PCIe as x1, x2, x4, x8, or x16. This should remain at 3'b011
6:4	0x3	FUSE_PCIE_WIDTH_T0C0: Configure PCIe as x1, x2, x4, x8, or x16. This should remain at 3'b011
2	0x1	FUSE_PCIE_T0_GEN2_DIS: Disable Gen 2 capability of PCIe.
1	0x1	FUSE_PCIE_SLI_DIS: Disable SLI capability for the GPU. This should remain on.

34.5.25 AFI_PMU_0

PMU Interface

Offset: 0x108 | Read/Write: R/W | Reset: 0x0XXXX000 (0bxxxxxxxxxxxxxxxxxxxx00000000xxx0)

Bit	R/W	Reset	Description
24	RO	X	CTLR_T0_C1_2PMU_TOG: PMU toggle response from PCIe

Bit	R/W	Reset	Description
23:20	RO	X	CTLR_T0_C1_2PMU_STATUS: PMU Status
16	RO	X	CTLR_T0_C0_2PMU_TOG: PMU toggle response from PCIe
15:12	RO	X	CTLR_T0_C0_2PMU_STATUS: PMU Status
11:8	RW	0x0	PMU2CTLR_T0_C1_LOAD_INDICATOR_SCALE: PMU Load Indicator Scale for T0C1
7:4	RW	0x0	PMU2CTLR_T0_C0_LOAD_INDICATOR_SCALE: PMU Load Indicator Scale for T0C0
0	RW	0x0	PMU2ALL_LI_UPDATE_FAST_TOG: PMU Load Indicator Enable. This is used for wall-plug applications and should remain off.

34.5.26 AFI_PCIE_CLK_CONFIG_STATUS_0

PCIE2 CLK Config/Status

Offset: 0x10c | Read/Write: R/W | Reset: 0x0XXXXX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	R/W	Reset	Description
27:24	RO	X	PCIE2CLK_TMS0GRP2_PAD_MACRO_CLK_SEL: Clock select to pad macro. This should remain at 0.
23:20	RO	X	PCIE2CLK_TMS0GRP1_PAD_MACRO_CLK_SEL: Clock select to pad macro. This should remain at 0.
19:16	RO	X	PCIE2CLK_TMS0GRP0_PAD_MACRO_CLK_SEL: Clock select to pad macro. This should remain at 0.
13	RO	X	PCIE2CLK_TMS0C1_DIS_XTXCLK1X: Request to gate T0C1 XTXCLK1X when in low power mode.
12	RO	X	PCIE2CLK_TMS0C0_DIS_XTXCLK1X: Request to gate T0C0 XTXCLK1X when in low power mode.
11	RO	X	PCIE2CLK_TMS0_CLAMP_CLK_L1: Request to gate TMS/FPCI clocks when in low power mode.
10	RO	X	PCIE2CLK_TMS0C1_SEL_XTXCLK1X_GEN2: Request to select Gen2 speed clock (500 MHz) for T0C1 XTXCLK1X. This is generated when register settings for PCIe2 specify Gen2 speed clocks.
9	RO	X	PCIE2CLK_TMS0C0_SEL_XTXCLK1X_GEN2: Request to select Gen2 speed clock (500 MHz) for T0C0 XTXCLK1X. This is generated when register settings for PCIe2 specify Gen2 speed clocks. This should remain low.
8	RO	X	PCIE2CLK_TMS0_SEL_XCLK_GEN2: Request to select Gen2 speed clock (500 MHz) for XCLK. This is generated when register settings for PCIe2 specify Gen2 speed clocks. This should remain low.
4	RW	0x0	CLK2PCIE_TMS0C1_OFF_XTXCLK1X: Acknowledge to disable T0C1 XTXCLK1X request. Used for clock gating.
3	RW	0x0	CLK2PCIE_TMS0C1_RDY_XTXCLK1X_GEN2: Acknowledge to select T0C1 XTXCLK1X Gen2 request. This should remain low.
2	RW	0x0	CLK2PCIE_TMS0C0_OFF_XTXCLK1X: Acknowledge to disable T0C0 XTXCLK1X request. Used for clock gating.
1	RW	0x0	CLK2PCIE_TMS0C0_RDY_XTXCLK1X_GEN2: Acknowledge to select T0C0 XTXCLK1X Gen2 request. This should remain low.
0	RW	0x0	CLK2PCIE_TMS0_RDY_XCLK_GEN2: Acknowledge to select XCLK Gen2 request. This should remain low.

34.5.27 AFI_PEX0_CTRL_0

PCIe PHY and Sideband Signal Interface

Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00x00)

Bit	Reset	Description
4	0x0	PEX0_REFCLK_OVERRIDE_EN: PEX0 enable to override refclk to be enabled always if PEX0_REFCLK_EN is set
3	0x0	PEX0_REFCLK_EN: PEX0 enable to clkout pad
1	0x0	PEX0_CLKREQ_EN: PEX0 enable clkreq to control clkout pad
0	0x0	PEX0_RST_L: PEX0 external pe0_rst_l register

34.5.28 AFI_PEX0_STATUS_0

Offset: 0x114 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	PEX0_CLKREQ_L: Status of the PEX0 pe0_clkreq_l input

34.5.29 AFI_PEX1_CTRL_0

Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00x00)

Bit	Reset	Description
4	0x0	PEX1_REFCLK_OVERRIDE_EN: PEX1 enable to override refclk to be enabled always if PEX1_REFCLK_EN is set
3	0x0	PEX1_REFCLK_EN: PEX1 enable to clkout pad
1	0x0	PEX1_CLKREQ_EN: PEX1 enable clkreq to control clkout pad
0	0x0	PEX1_RST_L: PEX1 external pe1_rst_l register

34.5.30 AFI_PEX1_STATUS_0

Offset: 0x11c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	PEX1_CLKREQ_L: Status of the PEX1 pe1_clkreq_l input

34.5.31 AFI_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x158 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	NUM_INITIATOR_ISO_PW_RESP_PEND_T0C1: Number of pending initiator ISO PW responses for controller 1
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND_T0C0:Number of pending initiator ISO PW responses for controller 0

34.5.32 AFI_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x15c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	NUM_INITIATOR_NISO_PW_RESP_PEND_T0C1: Number of pending initiator NISO PW responses for controller 1
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND_T0C0: Number of pending initiator NISO PW responses for controller 0

34.5.33 AFI_PLLE_CONTROL_0

PLLE Controls

Offset: 0x160 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxx00xxxxxx10)

Bit	Reset	Description
9	0x0	BYPASS_PADS2PLLE_CONTROL: Overrides PCIe PADS CLKREQ control of the PLLE.
8	0x0	BYPASS_PCIE2PLLE_CONTROL: Overrides PCIe2 CLOCK CLAMP control of the PLLE.
1	0x1	PADS2PLLE_CONTROL_EN: When active, all CLKREQs going inactive indicate DISABLE to the PLLE. When SATA, PCIe2, and PADS all indicate DISABLE to the PLLE, then the PLLE is disabled.
0	0x0	PCIE2PLLE_CONTROL_EN: When active, PCIE2 CLOCK CLAMP going active signal DISABLE to PLLE. When SATA, PCIe2, and PADS all indicate DISABLE to the PLLE, then the PLLE is disabled.

34.5.34 AFI_BUST_CONTROL_0

Bus Trace Module Controls

Offset: 0x164 | Read/Write: R/W | Reset: 0x00X00000 (0bxxxxxxxxxxx0xx00000000000000000)

Bit	R/W	Reset	Description
21	RO	X	PCIE_RXL_BUST_TRIG_OUT0_T0: Bus trigger from bus trace module in PCIe2

Bit	R/W	Reset	Description
20	RW	0x0	TRACE_EXTERNAL_START: Start signal to bus trace module in PCIe2
17:16	RW	0x0	PCIE_RXL_BUST_BUS_TRACE_MUX_SEL_T0: MUX select to bus trace module in PCIe2
15:0	RW	0x0	CHIP_ID: Chip ID to bus trace module in PCIe2

34.5.35 AFI_PEXBIAS_CTRL_0

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	PEX_BIAS_PWRD: PEX clock bias pad power down

34.5.36 AFI_P2PBOM_CTRL_0

Offset: 0x16c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	P2P_BASE:Peer-to-peer Bottom of Memory - connects to bits 39:20 of p2p_base_63_to_20 with the top bits tied to 0

34.5.37 AFI_P2PTOM_CTRL_0

Offset: 0x170 | Read/Write: R/W | Reset: 0x000003ff (0bxxxxxxxxxxxx00000000001111111111)

Bit	Reset	Description
19:0	0x3ff	P2P_LIMIT:Peer-to-peer Top of Memory - connects to bits 39:20 of p2p_limit_63_to_20 with the top bits tied to 0

34.5.38 AFI_CLKGATE_HYSTERESIS_0

Clock Gate Hysteresis

Offset: 0x174 | Read/Write: R/W | Reset: 0x00000014 (0bxxxxxxxxxxxxxxxxxxxxxxxx00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of AFI clock cycles to wait after clock gating criteria is met to disable the AFI/FPCI clocks

34.5.39 AFI_SPARE_REG0_0

Bit 31 is used as a Diagnostics bit to bypass both AXI target RAW and peer2peer RAW address comparison logic (default value is 1).

If 1, reads push all pending writes. Address range detection is bypassed.

If 0, reads only push pending writes that are in the same small address range.

Offset: 0x180 | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:0	-65536	IPFS_SPARE_REG: Spare Register.

34.5.40 AFI_A2F_UFPCI_CFG0_0

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000050 (0b00000000xxxxx0000000000001010000)

Bit	Reset	Description
31:24	0x0	STATIC_WAIT_IDLE_CNTR
18:16	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI1
15:12	0x0	STATIC_UFPCI_UFA_STARVE_CNTR_PRI0

Bit	Reset	Description
11:10	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI1
9:8	0x0	STATIC_UFPCI_RR_BURST_SZ_PRI0
7	0x0	STATIC_WAIT_CLAMP_EN
6	0x1	STATIC_UFPCI_UFA_DYN_BLOCK_EN
5	0x0	STATIC_UFPCI_UFA_BLK_COHERENT
4:2	0x4	STATIC_UFPCI_BLOCK_CMD_THRESHOLD
1	0x0	STATIC_CYA_UFA_ARB
0	0x0	STATIC_CYA_BACK2BACK_UPSTREAM_BLOCK

34.5.41 AFI_A2F_UFPCI_CFG1_0

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	STATIC_WAIT_UNCLAMP_CNTR

34.5.42 AFI_MSG_0

Reserved.

Offset: 0x190 | Read/Write: R/W | Reset: 0xXXX0XXX0 (0b000xxxxx0xxxxxxx000xxxxx0xxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x0	TMS0C12SM_MSG_ERR_FATAL:PCIE FATAL ERROR MSG
30	RW	0x0	TMS0C12SM_MSG_ERR_NONFATAL:PCIE NONFATAL ERROR MSG
29	RW	0x0	TMS0C12SM_MSG_ERR_COR:PCIE CORRECTABLE ERROR MSG
28	RO	X	TMS0C12SM_MSG_RP_INT:PCIE ROOT PORT INTERRUPT
27	RO	X	TMS0C12SM_MSG_INTD:PCIE INTERRUPT D MSG
26	RO	X	TMS0C12SM_MSG_INTC:PCIE INTERRUPT C MSG
25	RO	X	TMS0C12SM_MSG_INTB:PCIE INTERRUPT B MSG
24	RO	X	TMS0C12SM_MSG_INTA:PCIE INTERRUPT A MSG
23	RW	0x0	TMS0C12SM_MSG_HOTPLUG_SCI:Pink HOTPLUG MSG
20	RO	X	TMS0C12SM_PM_PME:PCIE Endpoint PME message
15	RW	0x0	TMS0C02SM_MSG_ERR_FATAL:PCIE FATAL ERROR MSG
14	RW	0x0	TMS0C02SM_MSG_ERR_NONFATAL:PCIE NONFATAL ERROR MSG
13	RW	0x0	TMS0C02SM_MSG_ERR_COR:PCIE CORRECTABLE ERROR MSG
12	RO	X	TMS0C02SM_MSG_RP_INT:PCIE ROOT PORT INTERRUPT
11	RO	X	TMS0C02SM_MSG_INTD:PCIE INTERRUPT D MSG
10	RO	X	TMS0C02SM_MSG_INTC:PCIE INTERRUPT C MSG
9	RO	X	TMS0C02SM_MSG_INTB:PCIE INTERRUPT B MSG
8	RO	X	TMS0C02SM_MSG_INTA:PCIE INTERRUPT A MSG
7	RW	0x0	TMS0C02SM_MSG_HOTPLUG_SCI:Pink HOTPLUG MSG
4	RO	X	TMS0C02SM_PM_PME:PCIE Endpoint PME message

34.5.43 AFI_AFI_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control (where applicable) and Clock Gating Control Register

Note: *The FIFO timing aspects of this register are no longer supported, but retained for software compatibility*

The clock override/ovr_mode fields of this register control the second-level clock gating for the client and MC side of the MCCIF. All clock gating is enabled by default.

A '1' written to the rclk/wclk override field will result in one of the following:

- With wclk/rclk override mode = LEGACY, the clock reverts to legacy mode of operation where the clock is on whenever the client clock is enabled.
- With wclk/rclk override mode = ON, the clock is always on inside MCCIF and PC. A '1' written to the cclk override field keeps client's clock always on inside MCCIF.

Offset: 0x7cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000xxxxxxxxxxxx0000)

Bit	Reset	Description
20	LEGACY	AFI_RCLK_OVR_MODE: 0 = LEGACY 1 = ON
19	LEGACY	AFI_WCLK_OVR_MODE: 0 = LEGACY 1 = ON
18	0x0	AFI_CCLK_OVERRIDE
17	0x0	AFI_RCLK_OVERRIDE
16	0x0	AFI_WCLK_OVERRIDE
3	DISABLE	AFI_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	AFI_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	AFI_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	AFI_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

34.5.44 AFI_ORDERING_RULES_0

This register is reserved.

Offset: 0x7d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000x)

Bit	Reset	Description
3	0x0	UPSTREAM_MSIAW
2	0x0	UPSTREAM_RESPAW
1	0x0	UPSTREAM_RAW

CHAPTER 35: I2C CONTROLLER

The I^2C controller (I2C) implements an I^2C 3.0 specification-compliant I^2C master and slave controller. The I^2C controller supports multiple masters and slaves. It supports Standard mode (up to 100 Kbits/s), Fast mode (up to 400 Kbits/s), and Fast mode plus (Fm+, up to 1 Mbits/s).

Tegra® X1 devices have six instances of this controller. All six instances have identical I2C master functionality. There are also three additional I2C instances in the TSEC, CL-DVFS, and VI modules.

The I^2C controller supports DMA for the Master controller over the APB bus. There is no DMA support for the Slave controller. The I^2C controller also supports packet mode transfers where the data to be transferred is encapsulated in a predefined packet format as payload and sent to the I^2C controller over the APB bus. The header of the packet specifies the type of operation to be performed, the size and other parameters (described in detail in subsequent sections).

35.1 Features

- Standard (up to 100 Kbits/s), Fast (up to 400 Kbits/s), and Fm+ (up to 1 Mbits/s) modes of operation.
- Single master
- Clock stretching by the slave
- One to eight-word burst data transfers in DMA mode for the Master controller
- 4 Kbyte transfers in packet mode. Can be extended beyond 4 Kbytes by breaking up transfers into multiple packets (in both DMA and non-DMA modes)
- 7-bit or 10-bit addressing transactions
- Master is capable of data transfers to/from two or more slaves consecutively with a repeated-start condition
- Bus clear operations to address SDA driven LOW issues
- General call addressing
- Recognition and transfer of data to peripherals that do not send an acknowledge (ACK)
- Master supports DMA for packet mode transfers

Note: *Neither multi-master mode nor high-speed (HS) operation mode are supported. References to those below should be ignored and corresponding register bits should be treated as reserved.*

35.2 Clocking

The I^2C controller has three clock domains:

- The Host interface runs on the APB clock (pclk). The APB clock is derived from the system clock and can be 1.0, ½, or ¼ times the system clock (sclk).
- The I^2C interface controller clock runs on a frequency up to 136 MHz (maximum). The 136 MHz clock is derived from PLLP. Even though you can mux between PLLP, PLLC, PLLM, and OSC clocks, PLLP is always selected and used in normal operation.
- The time-out logic runs on a 32 kHz clock.

35.3 Functionality

The I^2C Master can be programmed for either a single slave transaction or a two-slave transaction. The two-slave transaction is generally useful in a random read from an external device. When reading from an external device, the random read-address needs to be set with an initial dummy-write to the device, followed by a repeated-start and a read transaction to the same device.

35.3.1 Bus Clear Master Operation

Bus clear logic helps to resolve I2C bus hang issues. If the data line (SDA) is permanently stuck low because a slave device is pulling it low continuously for some unknown reason, the I2C master loses the bus arbitration and stops driving the I2C bus. In this case, software can use the Bus Clear master to get the SDA line released by sending clock pulses on the SCL line. Per the I2C specification, the device that held the bus Low should release it sometime within 9 clock pulses. But if the device needs more clock pulses than the default 9 pulses, software has an option to do so by programming a configurable register bit field (I2C_I2C_BUS_CLEAR_CONFIG_0[BC_SCLK_THRESHOLD]) to the required number of clock pulses.

35.3.1.1 Bus Clear Programming Procedure

The Bus Clear operation starts with an ARB_LOST notification from the I^2C controller. The ARB_LOST notification could be due to many reasons. When software initiates Tegra-I2C for a data transfer at this time, the master could back off if it does not win the bus in the bus-arbitration procedure. Other cases for ARB_LOST are programming issues such as controller register configuration, pinmuxing, and pin tri-state clear.

If it was determined that ARB_LOST notification is due to a slave device continuously pulling the SDA line low, the bus clear operation can be initiated to recover the bus with the following steps:

1. Reset the I2C controller using module reset after the ARB_LOST notification from the controller.
2. Program the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_SCLK_THRESHOLD] bit field to the required number of clock pulses.
3. Select the STOP or NO_STOP condition using the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_STOP_COND] bit field.
4. Program I2C_I2C_BUS_CLEAR_CONFIG_0[BC_TERMINATE] as needed. Always use the BC_TERMINATE = IMMEDIATE option.

Note: *If BC_TERMINATE =THRESHOLD settings are required to be used in any application with BC_STOP_COND = STOP settings, perform BusClear one more time. In the second BusClear iteration, use the BC_TERMINATE = IMMEDIATE settings with BC_STOP_COND = STOP settings.*

---

5. Set the I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] register bit field to 1, and wait until this bit field is auto-cleared to zero by hardware.
6. Set the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_ENABLE] bit field to start the bus clear operation. Hardware auto clears this bit after the bus clear operation is done.
7. Once enabled, the bus clear logic starts sending clock pulses on the SCL line.
8. Based on BC_TERMINATE settings, the Bus Clear logic:
   - IMMEDIATE: Sends clock pulses until SDA is released or the threshold count is reached, whichever is earlier or
   - THRESHOLD: Sends the threshold number of clock pulses irrespective of SDA line release status (released before threshold is reached or not-released with threshold count reached)
9. If SDA is released within the programmed number of clock pulses:
   - The Bus Clear logic terminates the transaction with STOP/NO-STOP condition based on the BC_STOP_COND settings.
   - Then it sets the bus_clear operation status in the I2C_I2C_BUS_CLEAR_STATUS_0[BC_STATUS] register bit field.

- It sets the BUS_CLEAR DONE bit field in I2C_INTERRUPT_STATUS_REGISTER_0 and also interrupts the system if the corresponding interrupt enable bit is set in I2C_INTERRUPT_MASK_REGISTER_0. So software has to wait until it receives the BUS_CLEAR_DONE interrupt if the interrupt mechanism is used or the BUS_CLEAR_DONE status bit is set if the status-polling mechanism used before going for a BUS CLEAR operation successful/fail status check in I2C_I2C_BUS_CLEAR_STATUS_0[BC_STATUS].
- After receiving BUS_CLEAR_DONE interrupt, check if I2C_I2C_BUS_CLEAR_CONFIG_0[BC_ENABLE] bit is auto-cleared by hardware. If this bit is not cleared, it means, I2C clock settings are not correct. Fix the clocks programming.

35.3.2 Low-Power Operation

I^2C is a low-speed serial interface with only two bus lines (SCL, SDA) required. So from the I^2C specification side, there is no low-power operation specifically defined for the interface. But to save the power, use the lowest possible speed, if performance is not needed. The following bus speeds are supported by I^2C:

- Standard mode (Sm), with a bit rate up to 100 kbit/s
- Fast mode (Fm), with a bit rate up to 400 kbit/s
- Fast mode Plus (Fm+), with a bit rate up to 1 Mbit/s

35.3.2.1 Low Power Programming

Program the CLK_SOURCE_I2C register (for example: I2C1instance clock source address 0x60006124) and the I2C_CLK_DIVISOR registers such that the required data rate can be achieved at the minimum possible module clock so that power can be saved.

35.3.2.2 IDLE Power Programming

In single Master bus configurations, when the module is not used or interface is idle, disable the module clock by clearing the CLK_ENB_I2C bits in the CAR module.

For example, the I2C1 clock enable bit is CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0[CLK_ENB_I2C1]. Software can clear this bit when it is seen that no transfers are needed from the I2C1 instance.

35.4 PWR-I2C Bus Sharing by the I2C5 and CLDVFS-I2C Controllers

It is possible for the I2C5 and CLDVFS-I2C controllers to try to access the PWR-I2C bus simultaneously in an application. To avoid bus conflicts, the CL-DVFS module provides a bus arbitration mechanism that grants the interface bus based on the requests to either the CLDVFS-I2C or I2C5 controller in round-robin fashion. If software intends to disable the CL-DVFS clock for some reason (for example, power saving), it has to put the CL-DVFS module in RESET by programming its reset to 1. So the I2C5 controller always gets the bus grant this way and the bus does not hang.

For the I2C5 controller to work properly, one of the following should occur:

- CLDVFS is in RESET

or

- CLDVFS is out of reset and its clocks are running

The I2C5 controller will not work if the CL-DVFS module is out of reset and the clocks to the CL-DVFS module are disabled. This is because the interface bus grant arbiter in the CL-DVFS cannot work because its clock is no longer toggling.

35.5 Software Interfaces

A typical peripheral controller might require the following steps to communicate or control an external peripheral:

1. Set up the registers
2. Program the ‘go’ bit to start the interaction with the external peripheral

3. The peripheral controller collects the response from the peripheral and interrupts software
4. Software reads the response

35.5.1 Packet Flow

Information exchange between the hardware peripheral controller and software as described above can be classified into:

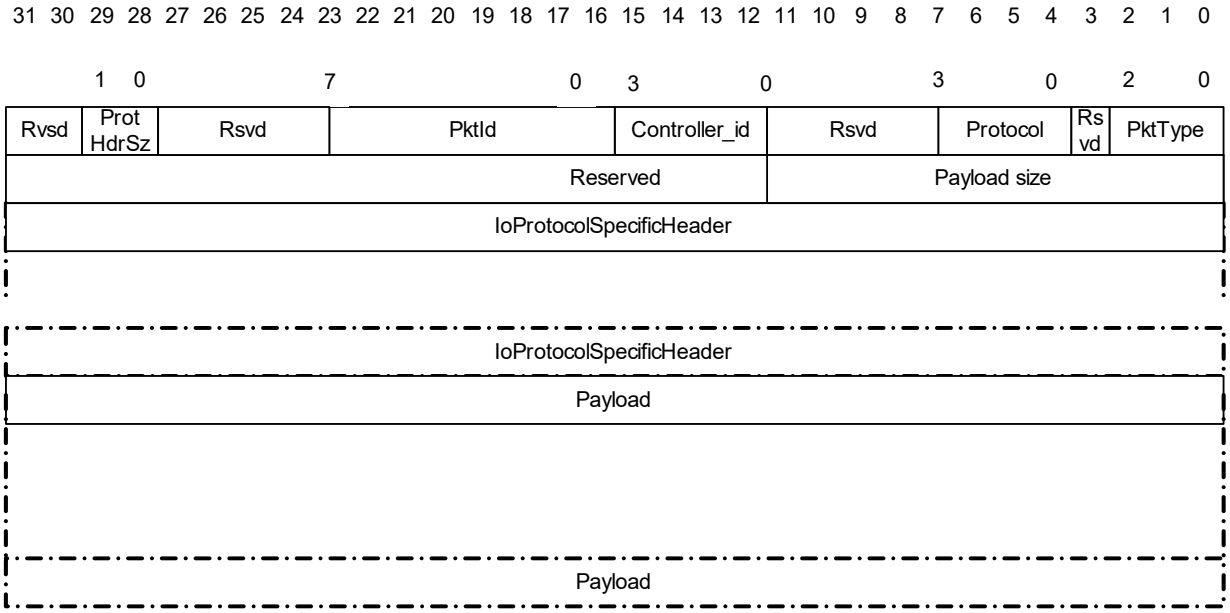
- 'Request' flow which can represent the register writes
- 'Response' flow for peripheral controller responses sent back to software, and
- 'Interrupt' flow for out-of-band handshakes between hardware and software

Packets within a flow can be exchanged in a manner that requires full intervention from the CPU (PIO mode) or least intervention from the CPU (DMA mode). The I2C I/O packet structure is described below:

35.5.2 I/O Packet Format

An I/O packet consists of a header and payload. The header consists of two parts: two words of generic header and 1-4 words of protocol-specific header.

Figure 159: I/O Packet



35.5.2.1 I/O Packet Word 0

Word 0 of the I/O packet contains the Generic Header Word 0. The contents of generic header word 0 are described in the table below.

Table 219: Generic I/O Header Word 0

Bits	Field Value	Description
31:30	Reserved	Reserved
29:28	ProtHdrSz	Size of the protocol-specific header in words 0 = 1 words 1 = 2 words 2 = 3 words 3 = 4 words For I2C, ProtHdrSz = 0 for request packets and 1 for response packets
27:24	Reserved	Reserved

Table 219: Generic I/O Header Word 0

Bits	Field Value	Description
23:16	PktId	PktId is an 8-bit field that identifies the packet. A peripheral controller might choose to log the packet ID for a packet that causes errors
15:12	Controller ID	There might be multiple controllers supporting any format. For example, there might be 4 I2C controllers. The Controller ID field specifies the controller for which this packet is destined.
11:8	Reserved	Reserved
7:4	Protocol	The protocol is indicated in this field 0 = reserved 1= I2C 2-15 = reserved
3	Reserved	Reserved
2:0	PktType	Packet type information 0 = request 1 = response 2 = interrupt 3 = stop 4-7 = reserved

35.5.2.2 I/O Packet Word 1

Word 1 of the I/O packet contains Generic Header Word 1. The contents of Generic Header Word 1 are described in the table below.

Table 220: Generic I/O Header Word 1

Bits	Field Value	Description
31:12	Reserved	Reserved
11:0	PayloadSize	Payload size in bytes

Note: *Payload Size: For request packets that transmit data, the total packet size is payload_size + 8 bytes of generic header + protocol-specific header. However, for request packets with a read command, the payload size indicates the amount of data to be received. Hence the packet size = 8 bytes of generic header + protocol-specific header, because there is no payload in this packet, just the receive command.*

35.5.3 Transferring Packets

Packets can be transferred using PIO or DMA modes. A peripheral controller can choose to implement one or both modes. The header is a minimum of three words with the first two words reserved for packet information and the third word reserved for protocol-specific requirements. A minimum of four words is needed to transfer one byte to the external peripheral.

35.5.4 Protocol-Specific Header

Depending on the implementation of the peripheral controller and the interface protocol, the header words can be defined. No restriction is placed on the organization of the fields or the number of protocol-specific words (a minimum of one word and a maximum of four words) that a particular implementation chooses.

I2C has one word with the Protocol-Specific Header. The contents of the I2C protocol-specific header are described in the next section.

35.5.5 I2C Master Packet Protocol Specific Headers

The I2C master supports two kinds of packets.

- Transmit packets that flow from Host to Controller.

- Receive data packets that flow from Controller to Host.

Table 221: I2C Master Transmit Packet Header

Bit	Name	Description
31:26	Reserved	Reserved
25	RESP_PKT_FREQ	Reserved Program 0.
24	RESP_PKT_ENABLE	Reserved. Program 0.
23	Reserved	Reserved
22	HS_MODE	Enable High speed mode (3.4 MHz operation)
21	CONTINUE_ON_NACK	Enable mode to handle devices that do not generate ACK upon the reception of a byte.
20	SEND_START_BYTE	1 = Send a start byte at the beginning of the transaction
19	READ/WRITE	1= READ
18	Address mode	1=10-bit mode 0 = 7-bit mode
17	IE	Generate interrupt upon packet completion
16	REPEAT_START/STOP	1 = Put a repeat start condition on the bus (to continue transaction) 0 = Put a stop condition on the bus
15	ContinueXfer	This bit overrides the REPEAT_START/STOP command above. 0: Introduce repeat start or stop condition based on bit 16. 1: Continue with current transfer without stop or repeat start condition.
14:12	HS_MASTER_ADDR	High Speed mode Master code
11:10	Reserved	Reserved
9:0	SLAVE_ADDR	Slave address. Bit 0 is ignored for 7-bit addressing, but should always match bit 19 (READ/WRITE).

35.5.6 I2C Master Speed Modes Support in Normal Mode vs Packet Mode

Table 222: Speed Modes in Normal/Packet Mode

Normal Mode	Packet Mode
Standard Mode (SM, 100KHz) Fast Mode (FM, 400KHz) Fast Mode Plus (FM+, 1MHz)	Standard Mode (SM, 100KHz) Fast Mode (FM, 400KHz) Fast Mode Plus (FM+, 1MHz)

35.6 Programming Guidelines

35.6.1 I2C Frequency Divisor Register

The FREQUENCY DIVISOR register (CLK_SOURCE_I2C register, whose address is: 0x60006124 for I2C1) must be programmed as a function of the CLK_SOURCE selected for I2C as follows:

Standard/Fast-mode/Fm+:

$$SCL\ Rate\ = \frac{I2Cx\_CLK\_SRC}{(I2Cx\_CLK\_DIVISOR + 1) * (TLOW + THIGH + 2 + Round\_Trip\_Delay\_In\_DivClk\_cycles) * (I2C\_CLK\_DIVISOR\_STD\_FAST\_MODE + 1)}$$

Where x = 1, 2, 3… 6

Note: *I2C_CLK_DIVISOR_STD_FAST_MODE settings should be such that it meets the following the requirement.*

$$\frac{(I2Cx\_CLK\_DIVISOR + 1) * (I2C\_CLK\_DIVISOR\_STD\_FAST\_MODE + 1) * 1000}{I2Cx\_CLK\_SRC\ in\ MHz} > \frac{Minimum\ SCL\ LowPeriod\ in\ nanosec}{5}$$

Table 223: Standard Mode Examples: SCL Low Period (minimum) = 4700 ns

I2C_CLK_SRC	I2C_CLK_SRC Frequency (MHz)	I2C_CLK_DIVISOR	I2C_CLK_DIVISOR_STD_FAST_MODE	Time Period (ns) = ((I2C_CLK_DIVISOR+1)*(I2C_CLK_DIVISOR_STD_FAST_MODE+1)*1000)/I2C_CLK_SRC	Time Period greater than minimum SCL Low Period/5 (4700/5 = 940ns) ?
PLLP_OUT0	408	2	128	948.5294118	YES
PLLP_OUT0	408	19	19	980.3921569	YES
CLK_M	19.2	0	23	1250	YES

Table 224: Fast Mode Examples: SCL Low Period (minimum) = 1300 ns

I2C_CLK_SRC	I2C_CLK_SRC Frequency (MHz)	I2C_CLK_DIVISOR	I2C_CLK_DIVISOR_STD_FAST_MODE	Time Period (ns) = ((I2C_CLK_DIVISOR+1)*(I2C_CLK_DIVISOR_STD_FAST_MODE+1)*1000)/I2C_CLK_SRC	Time Period greater than minimum SCL Low Period/5 (1300/5 = 260ns) ?
PLLP_OUT0	408	2	13	102.9411765	YES
CLK_M	38.4	0	3	104.1666667	YES

Table 225: Fast Mode Plus Examples: SCL Low Period (minimum) = 500 ns

I2C_CLK_SRC	I2C_CLK_SRC Frequency (MHz)	I2C_CLK_DIVISOR	I2C_CLK_DIVISOR_STD_FAST_MODE	Time Period (ns) = ((I2C_CLK_DIVISOR+1)*(I2C_CLK_DIVISOR_STD_FAST_MODE+1)*1000)/I2C_CLK_SRC	Time Period greater than minimum SCL Low Period/5 (1300/5 = 260ns) ?
PLLP_OUT0	408	2	13	102.9411765	YES
CLK_M	38.4	0	3	104.1666667	YES

Table 226: Example Settings for Various I2C Speeds

Source	PLLP_OUT0	I2C Source Div (Programmed value is 1 minus always)	(Tlow+1)+ (Thigh+1)	SM/FM Div (Programmed Value is 1 minus always)	I2C Frequency	Debounce
PLLP	408 MHz	3 (2)	((4+1) + (2+1)) = 8	17 (16)	1 MHz (FM+)	0
PLLP	408 MHz	5 (4)	((4+1) + (2+1)) = 8	26 (25)	392 kHz (FM)	2
PLLP	408 MHz	20 (19)	((4+1) + (3+1)) = 9	23 (22)	98.5 kHz (SM)	2

Debounce will impact the I2C interface data rate in the following way.

For example, if CLK_SOURCE = 408 MHz and Source Div = 5:

I2C_Clock frequency = 408/5 = 81.6 MHz ➔ I2C clock period = 12.25 ns

Chip internal delay = 10 ns, Load_Delay = 55 ns, Debounce = 2, Synchronizer Delay = 3

Round_Trip_Delay = (10 ns + 55 ns + 9*12.25 ns) = ~175.25 ns

FM+ speed case: STD_FM_Div = 10

STD_FM_Div * I2C_Clk_Period = 10*12.255 ns = 122.55 ns

Round_Trip_Delay in I2C_Div cycles = Round_Trip_Delay/(STD_FM_Div * I2C_Clk_Period) = 1. (Rounded down to the nearest integer.)

So I2C SCL rate = $\frac{408MHz}{5*(8+1)*10}$ = 906 KHz

If Debounce is disabled, the Round_Trip_delay (in I2C_Div cycles) becomes zero (rounded down).

So I2C SCL rate = $\frac{408MHz}{5*(8)*10}$ = 1002 KHz = 1.02 MHz

Table 227: I2C Data Rate Variation with Debounce

I2C Mode	Clock Source	Source Clock Frequency	I2C Source Divisor (Programmed value is always 1 minus always)	SM/FM Divisor (Programmed value is always 1 minus always)	Debounce Value	I2C SCL Frequency
FM+	PLLP_OUT0	408 MHz	5 (4)	10 (9)	0	1016 kHz
			5 (4)	10 (9)	Range: 5 to 1	906 kHz
			5 (4)	10 (9)	Range: 7 to 6	816 kHz
FM	PLLP_OUT0	408 MHz	5 (4)	26 (25)	Range: 7 to 0	392 kHz
SM	PLLP_OUT0	408 MHz	20 (19)	26 (25)	Range: 7 to 0	98 kHz

The clock enable (bit 12 of the CLK_OUT_ENB.L register) must also be given to the I2C controller, before any of the registers are written.

35.6.2 I2C Command ADDR Registers

These registers, I2C_CMD_ADDR0 and I2C_CMD_ADDR1, contain the address of the slave with which a transaction is intended. The I^2C Master Controller can be programmed to transact with both 7-bit and 10-bit addressed slaves. Bit [0] of the I2C_CNFG register determines the size of the slave address to be transacted. If a 7-bit or a 10-bit slave address is chosen, the respective address is taken from I2C_CMD_ADDR0 [9:0], and the Read/Write command is taken from bits 6 and 7 of the I2C_CNFG Register. In a 2 Slave configuration, the slave 1 address (7-bit or 10-bit) is taken from I2C_CMD_ADDR0 [9:0] and Slave 2 address (7-bit or 10-bit) from I2C_CMD_ADDR1 [9:0]. The transfer length is the same in 2 slave operation and is taken from the I2C_CNFG [LENGTH] bit field. In 2-slave operation, the maximum possible transfer size is 4 bytes.

35.6.3 I2C Data Registers

There are two data registers, I2C_CMD_DATA1and I2C_CMD_DATA2, each with 32-bit length, which are to be loaded with the data to be transmitted or received (I2C_SL_RCVD register).

35.6.4 I2C Configuration Register

This register is to be configured with the following data:

- Number of bytes to be transmitted or received
- Select for 7-bit or 10-bit slave address
- Program to send a Start Byte
- Single or two slave operations

- Support of NOACK from an external peripheral

The I2C_CNGF [9] bit is used to issue a “Begin-Transaction” command to the I^2C Master Controller. This bit is auto cleared by the hardware, and other bits of the register are masked for writes, when this bit is programmed to be one. Hence, the firmware should first configure all the other registers and bits [8:0] of the I2C_CNFG register before the I2C_CNFG [9] bit is programmed to one.

35.6.5 I2C Configuration Load Register

This CONFIG load register transfers the software programmed configuration in the I2C registers to hardware internal registers used in the logic. It has three bit fields:

- MSTR_CONFIG_LOAD for regular master and Bus Clear master logic
- SLV_CONFIG_LOAD for slave controller logic
- TIMEOUT_CONFIG_LOAD for SMBUS timeout logic.

Software must set these fields to 1 for the actual register configuration to take effect. Thus software is programming only shadow registers through regular configuration. When these load_config bit fields are set to 1, it causes the regular/shadow registers configuration to be transferred to the hardware internal active registers. So software has to set these bit fields at the end of all regular register configurations. However, in normal or non-packet mode, these bit fields should be set to 1 before the I2C_I2C_CNFG_0[SEND] bit is set to 1. The I2C_I2C_CNFG_0[SEND] bit triggers a transfer on the I2C bus and should be programmed at the end of the entire configuration. These config_load bits are hardware auto-clear bits. Hardware clears these bit fields once the register configuration is moved to hardware internal active registers. So software must wait until these bits are auto-cleared before continuing with further programming.

Below are the register bit fields associated with each CONFIG_LOAD bit.

MSTR_CONFIG_LOAD needs to be set for the following registers:

- I2C_I2C_CNFG_0
- I2C_I2C_CMD_ADDR0_0
- I2C_I2C_CMD_ADDR1_0
- I2C_I2C_CMD_DATA1_0
- I2C_I2C_CMD_DATA2_0
- I2C_I2C_CLK_DIVISOR_REGISTER_0
- I2C_I2C_BUS_CLEAR_CONFIG_0
- I2C_I2C_INTERFACE_TIMING_0_0
- I2C_I2C_INTERFACE_TIMING_1_0
- I2C_I2C_HS_INTERFACE_TIMING_0_0
- I2C_I2C_HS_INTERFACE_TIMING_1_0

SLV_CONFIG_LOAD covers the following registers:

- I2C_I2C_SL_CNFG_0
- I2C_I2C_SL_ADDR1_0
- I2C_I2C_SL_ADDR2_0
- I2C_I2C_TLOW_SEXT_0[RST_SL_ON_TIMEOUT]
- I2C_I2C_SL_DELAY_COUNT_0

TIMEOUT_CONFIG_LOAD covers the following register bit fields:

- I2C_I2C_TLOW_SEXT_0[TIMEOUT]

- I2C_I2C_TLOW_SEXT_0[TLOW_SEXT]
- I2C_I2C_TLOW_SEXT_0[TLOW_MEXT]
- I2C_I2C_TLOW_SEXT_0[TIMEOUT_EN]
- I2C_I2C_TLOW_SEXT_0[TLOW_SEXT_EN]
- I2C_I2C_TLOW_SEXT_0[TLOW_MEXT_EN]

35.6.6 I2C Controller Status Register

This register (I2C_STATUS) gives the status of the I^2C Master operation. It includes the busy status of the I^2C Master controller and the completion status of the command executed.

35.6.7 I2C Interface Timing Registers

These registers provide flexibility to tune the I2C interface bus timing for Master controller data transfers. Separate bit fields are provided for non-HS mode (STD/FM/FM+) timing.

35.6.8 I2C Slave ADDR Registers

There are two slave ADDR registers that are used to configure the address of the internal slave, I2C_SL_ADDR1 and I2C_SL_ADDR2. The bit [0] of I2C_SL_ADDR2 determines the address-size of the internal slave. When this bit is programmed to zero, the 7-bit slave address is taken from I2C_SL_ADDR1 [6:0]. When this bit is programmed to one, the most significant two bits of the 10-bit slave address are taken from I2C_SL_ADDR2[2:1] and the least significant bits are taken from I2C_SL_ADDR1[7:0].

35.6.9 I2C Controller Slave Status Register

This register (I2C_SL_STATUS) contains the status of I2C Slave.

It has the following status bits:

1. Bit 3 contains Receive Interrupt.
2. Bit 2 contains New Transaction Received.
3. Bit 1 is for Direction of transfer.
4. Bit 4 contains END_TRANS bit. This bit is set when an I2C transfer is complete (as indicated by either a STOP or NoACK from master).
5. Bit 5 contains RST_SL. The byte received following the receipt of general call address is 0x06. Therefore, the programmable part of the slave address needs to be reset and reprogrammed.
6. Bit 6 contains REPROG_SL. The byte received following the receipt of general call address is 0x04. Therefore reprogram the programmable part of the slave address.
7. Bit 7 contains HW_MSTR_INT. When set, this bit indicates that master address has been transmitted after general call address.
8. Bits [14:8] contain the hardware master address. The contents of this field are meaningful only when bit 7 (HW_MSTR_INT) is set.

Note: *The I2C Slave Controller generates an interrupt at the completion of each data byte received in the byte mode of operation. In response to this interrupt, firmware must read the I2C_SL_STATUS register to confirm the cause of the interrupt and the direction of data transfer. If the interrupt is due to a data byte received (R/W=0), firmware must read the data register and write-1 to I2C_SL_STATUS [SL_IRQ] to clear the interrupt and get the SCL line released. If the interrupt is due to read operation (R/W=1), data should be written to I2C_SL_RCVD in order to release the SCL line from clock stretching. The I2C Controller will delay the release of the ACK handshake on the I2C bus until the SL_IRQ bit has been cleared by the firmware. SW has to write-1 to clear I2C_SL_STATUS register bits.*

I2C_SL_STATUS needs to be either polled or the I2C interrupt must be enabled for transfers to occur in slave mode.

If Tegra X1 slave is receiving data,

- I2C_SL_STATUS[SL_IRQ] bit being set indicates that byte has been received.
- Check the transfer direction from RNW field.
- Read data byte from I2C_SL_RCVD register.
- Write 1 to I2C_SL_STATUS [SL_IRQ] to clear SL_IRQ which causes the bus to be released.
- SL_IRQ is set again upon reception of next byte.

If Tegra X1 slave is transmitting data,

- I2C_SL_STATUS[SL_IRQ] bit being set indicates that address has been received.
- Check the transfer direction from the RNW field.
- Write 1 to I2C_SL_STATUS [SL_IRQ] to clear SL_IRQ.
- Write data byte to the I2C_SL_RCVD register. This write causes the bus to be released.
- SL_IRQ is set again upon transfer of the byte. Another byte can be written to continue the operation.

Slave stretches the SCL line in the following cases:

- Pending Interrupts from I2C_SL_STATUS register after a byte received in write or a data byte needs to be sent in read operation. Software has to clear them by writing 1 to the register bit fields which releases the SCL line.
- TX-FIFO is empty and no data is available to send to Master during read operation. Software to write data to TX-FIFO to get the SCL line released.
- RX-FIFO is full and no room is left in the FIFO to receive any more data from Master. Software to read the RX-FIFO to allow the SCL line released
- During address cycle, the SCL stretch happens after ACK bit. The stretch time includes the interrupt clear time and the SL_DELAY_COUNT. If it is read operation, the clock stretch is extended until data byte is written to the I2C_SL_RCVD register.
- During data phase, the SCL stretch happens before and after ACK bit. The stretch before ACK is for a duration of SL_DELAY_COUNT and the stretch time after ACK bit is the interrupt clear time plus the SL_DELAY_COUNT period. If it is the read operation, the clock stretch is extended until data byte is written to the I2C_SL_RCVD register.

35.6.10 Clock Gating Over Ride Enable Register

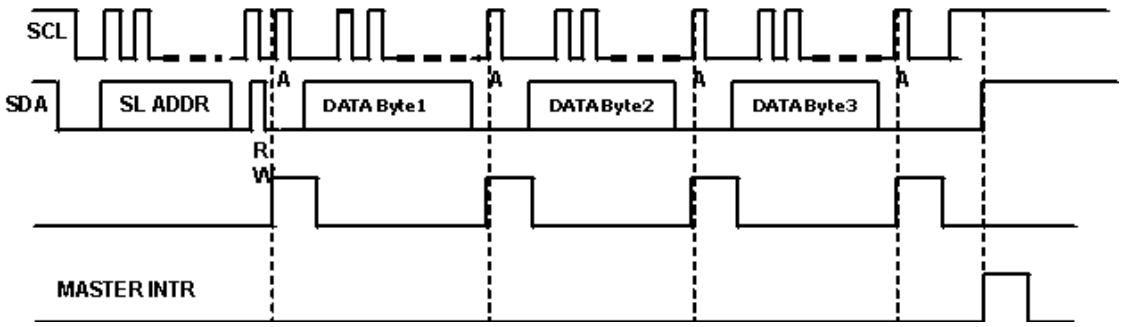
2nd level clock gating improves the power saving function. I2C implements SLCG in both the master and slave controller logic to save power. It is ENABLED by default. For any reason, if clock gating needs to be disabled, it can be done through the programmable I2C_CLKEN_OVERRIDE register. Separate bit fields are provided for master, bus clear master, and slave controllers.

35.6.11 Interrupt Generation in Non-Packet (Normal) Mode

In non-packet mode, the I2C controller generates interrupts upon the completion of transmission or reception of the specified number of bytes. A Master interrupt is generated when the number of bytes of the transaction, as programmed in the LENGTH field of the I2C_CNFG register, is complete.

A Slave-interrupt is generated at the transmission/reception of each byte of data including address byte by the internal slave as shown in figure below. Slave interrupts are to be cleared by the firmware as mentioned above.

Figure 160: Interrupt Generation: Master Reads from Internal Slave



35.7 Programming Guidelines for Master in Normal Mode

35.7.1 Programming Model for Master-Transmit

The following is a programming example for 7 byte writes from master to an external slave:

1. Program the I2C_CLK_DIVISOR register to get the required data rate based on I2C CLK_SOURCE register programming in CAR module.
2. Write slave address in I2C_CMD_ADDR0 register based on 7-bit/10-bit addressing mode.
3. Write first 4 bytes data in I2C_CMD_DATA1 register.
4. Write remaining 3 bytes in I2C_CMD_DATA2 register.
5. Program I2C_CNFG[DEBOUNCE_CNT] to the required value.
6. Set I2C_CNFG[A_MOD] to Seven or Ten_bit addressing based on slave device address width.
7. Program I2C_CNFG [LENGTH]. This bit field works in (n+1) fashion so for 7 bytes transfer, 6 needs to be programmed in LENGTH bit field.
8. Set I2C_CNFG [SLV2] = 0 as this is one slave access.
9. Set CMD1 = 0 for write operation.
10. Set NOACK to 0 or 1 based on Slave type.
11. Set MSTR_CONFIG_LOAD bit in I2C_CONFIG_LOAD register.
12. Wait until I2C_CONFIG_LOAD[MSTR_CONFIG_LOAD] is auto-cleared by hardware to 0.
13. Set I2C_CNFG [SEND] to 1 to begin write transaction on the interface.
14. Wait until the transaction is complete. For this, either wait until interrupt is received or I2C_STATUS[BUSY] bit becomes zero.
15. Check I2C_STATUS [CMD1_STAT] to see if the transaction is successful or NOACK from the slave for any of the bytes transferred.

35.7.2 Programming Model for Master-Receive

Read operation is divided into two steps based on random read or immediate read after a write. In a typical random read,

1. Write command: Send slave register/memory index first from which the data needs to be read.
2. Send read command to read the data.

The above operation can be done in two separate configurations or in a single configuration using Repeated Start in 2-slave configuration:

1. Write: Sending slave’s 1-byte register index first.

a. Write slave address in I2C_CMD_ADDR0 register based on 7-bit/10-bit addressing mode.
b. Write the slave internal register index (from which the data has to be read) in I2C_CMD_DATA1 register.
c. Program I2C_CNFG[DEBOUNCE_CNT] to the required value.
d. Set I2C_CNFG[A_MOD] to Seven or Ten_bit addressing based on slave device address width.
e. Write slave register index byte in I2C_CMD_DATA1 register.
f. Program I2C_CNFG[LENGTH] = 0 for 1 byte register index transfer.
g. Set I2C_CNFG[SLV2] = 0.
h. Set I2C_CNFG[CMD1] = 0 for write operation.
i. Set I2C_CNFG[NOACK] to 0 or 1 based on Slave type.
j. Set MSTR_CONFIG_LOAD bit in I2C_CONFIG_LOAD register.
k. Wait until I2C_CONFIG_LOAD[MSTR_CONFIG_LOAD] is auto-cleared by hardware to 0.
l. Set I2C_CNFG[SEND] to begin write transaction on the interface.
m. Wait until the transaction is complete. For this, either wait until interrupt is received or I2C_STATUS [BUSY] bit becomes zero.
n. Check I2C_STATUS[CMD1_STAT] to see if the transaction is successful or NOACK from the slave.

2. Read: Send read command.
   a. Program I2C_CNFG[LENGTH] = 6 for 7-bytes read.
   b. Set I2C_CNFG[SLV2] = 0 as this is one slave access.
   c. Set I2C_CNFG[CMD1] = 1 for read operation.
   d. Set MSTR_CONFIG_LOAD bit in I2C_CONFIG_LOAD register.
   e. Wait until I2C_CONFIG_LOAD[MSTR_CONFIG_LOAD] is auto-cleared by hardware to 0.
   f. Set I2C_CNFG[SEND] to begin write transaction on the interface.
   g. Wait until the transaction is complete. For this, either wait until interrupt is received or I2C_STATUS [BUSY] bit becomes zero.
   h. Check I2C_STATUS[CMD1_STAT] to see if the transaction is successful or NOACK from the slave.
   i. Read first 4 bytes data in I2C_CMD_DATA1 register.
   j. Read remaining 3 bytes in I2C_CMD_DATA2 register.
3. Read using repeated-start in 2-slave mode. Note that the maximum transfer size possible in 2-slave mode is 4 bytes.
   a. Write slave address in I2C_CMD_ADDR0 register with LSB bit set to “0”.
   b. Write slave address in I2C_CMD_ADDR1 register with LSB bit set to “1”.
   c. Write the slave internal register index byte (from which the data has to be read) in I2C_CMD_DATA1 register.
   d. Program I2C_CNFG[DEBOUNCE_CNT] to the required value as needed.
   e. Set I2C_CNFG[A_MOD] to Seven or Ten_bit addressing.
   f. Program I2C_CNFG[LENGTH] = 0 for 1 byte transfer.
   g. Set I2C_CNFG[SLV2] = 1.
   h. Set I2C_CNFG[CMD1] = 0 for write operation.
   i. Set I2C_CNFG[CMD2] = 1 for read operation.
   j. Set I2C_CNFG[NOACK] to 0 or 1 based on Slave type.
   k. Set MSTR_CONFIG_LOAD bit in I2C_CONFIG_LOAD register.
   l. Wait until I2C_CONFIG_LOAD[MSTR_CONFIG_LOAD] is auto-cleared by hardware to 0.

m. Finally set I2C_CNFG[SEND] to begin write transaction on the interface.

n. Wait until the transaction is complete. For this, either wait until interrupt is received or I2C_STATUS [BUSY] bit becomes zero.

o. Check I2C_STATUS [CMD1_STAT] to see if the transaction is successful or NOACK from the slave.

p. Read data from I2C_CMD_DATA2 register.

35.7.2.1 Example Transfers in Normal Mode

Write Example:

This programming example is for 7 byte writes from master to an external slave.

Program the I2C_CLK_DIVISOR register to get the required data rate based on I2C CLK_SOURCE register programming in CAR module

1. Write the slave address in the I2C_CMD_ADDR0 register based on 7-bit/10-bit addressing mode
2. Write the first 4 bytes of data in the I2C_CMD_DATA1 register
3. Write the remaining 3 bytes in I2C_CMD_DATA2 register
4. Program I2C_CNFG[DEBOUNCE_CNT] to the required value as needed
5. Set I2C_CNFG[A_MOD] to 7-bit or 10-bit addressing
6. Program I2C_CNFG [LENGTH]. This bit field works in (n+1) fashion; so for a 7-byte transfer, 6 needs to be programmed in the LENGTH bit field
7. Set I2C_CNFG [SLV2] = 0 as this is one slave access
8. Set CMD1 = 0 for write operation
9. Set NOACK to 0 or 1 based on Slave type
10. Set the MSTR_CONFIG_LOAD bit in the I2C_CONFIG_LOAD register
11. Set I2C_CNFG [SEND] to 1 to begin a write transaction on the interface
12. Wait until the transaction is complete, either until an interrupt is received or the I2C_STATUS[BUSY] bit becomes zero.
13. Check I2C_STATUS [CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave for any of the bytes transferred.

Read Example:

A read operation is divided into two steps based on random read or immediate read after a write. In a typical random read:

1. Write command: Send slave register/memory index first from which the data needs to be read.
2. Send read command to read the data. This operation can be done in two separate configurations or in a single configuration using Repeated Start in 2-slave configuration.

These steps are described in detail below:

- Write: Sending slave's 1-byte register index first.
  a. Write the slave address in I2C_CMD_ADDR0 register based on 7-bit/10-bit addressing mode.
  b. Write the slave's internal register index (from which the data has to be read) in the I2C_CMD_DATA1 register.
  c. Program I2C_CNFG[DEBOUNCE_CNT] to the required value as needed.
  d. Set I2C_CNFG[A_MOD] to 7-bit or 10-bit addressing.
  e. Write the slave register index byte in the I2C_CMD_DATA1 register.
  f. Program I2C_CNFG[LENGTH] = 0 for a 1 byte register index transfer.
  g. Set I2C_CNFG[SLV2] = 0.
  h. Set I2C_CNFG[CMD1] = 0 for write operation.

i. Set I2C_CNFG[NOACK] to 0 or 1 based on the Slave type.

j. Set the MSTR_CONFIG_LOAD bit in the I2C_CONFIG_LOAD register.

k. Set I2C_CNFG[SEND] to begin the write transaction on the interface.

l. Wait until the transaction is complete. (Wait until either an interrupt is received or the I2C_STATUS [BUSY] bit becomes zero).

m. Check I2C_STATUS[CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave.

- Read: Send read command.

a. Program I2C_CNFG[LENGTH] = 6 for 7-bytes read.

b. Set I2C_CNFG[SLV2] = 0 because this is one slave access.

c. Set I2C_CNFG[CMD1] = 1 for read operation.

d. Set the MSTR_CONFIG_LOAD bit in the I2C_CONFIG_LOAD register.

e. Set I2C_CNFG[SEND] to begin write transaction on the interface.

f. Wait until the transaction is complete. (Wait until either an interrupt is received or the I2C_STATUS [BUSY] bit becomes zero).

g. Check I2C_STATUS[CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave.

h. Read the first 4 bytes data in the I2C_CMD_DATA1 register.

i. Read the remaining 3 bytes in the I2C_CMD_DATA2 register.

- Read using repeated-start in 2-slave mode.

a. Write the slave address in the I2C_CMD_ADDR0 register with the LSB bit set to “0”.

b. Write the slave address in the I2C_CMD_ADDR1 register with the LSB bit set to “1”.

c. Write the slave internal register index byte (from which the data has to be read) in the I2C_CMD_DATA1 register.

d. Program I2C_CNFG[DEBOUNCE_CNT] to the required value as needed.

e. Set I2C_CNFG[A_MOD] to 7-bit or 10-bit addressing.

f. Program I2C_CNFG[LENGTH] = 0 for 1 byte transfer.

g. Set I2C_CNFG[SLV2] = 1.

h. Set I2C_CNFG[CMD1] = 0 for write operation.

i. Set I2C_CNFG[CMD2] = 1 for read operation.

j. Set I2C_CNFG[NOACK] to 0 or 1 based on the Slave type.

k. Set the MSTR_CONFIG_LOAD bit in the I2C_CONFIG_LOAD register.

l. Set I2C_CNFG[SEND] to begin the write transaction on the interface.

m. Wait until the transaction is complete. (Wait until either an interrupt is received or the I2C_STATUS [BUSY] bit becomes zero).

n. Check I2C_STATUS [CMD1_STAT] to see if the transaction is successful or there is a NOACK from the slave

o. Read data from the I2C_CMD_DATA2 register.

35.8 Programming Guidelines for Packet-Based Interface

35.8.1 Packet Based Interface Registers

The following registers are used for a packet based interface:

- I2C_I2C_TX_PACKET_FIFO_0
- I2C_I2C_RX_FIFO_0

- I2C_PACKET_TRANSFER_STATUS_0
- I2C_I2C_HS_INTERFACE_TIMING_0_0
- I2C_I2C_HS_INTERFACE_TIMING_1_0
- I2C_I2C_INTERFACE_TIMING_0_0
- I2C_I2C_INTERFACE_TIMING_1_0
- I2C_FIFO_CONTROL_0
- I2C_FIFO_STATUS_0
- I2C_INTERRUPT_MASK_REGISTER_0
- I2C_INTERRUPT_STATUS_REGISTER_0
- I2C_I2C_INTERRUPT_SOURCE_REGISTER_0
- I2C_I2C_INTERRUPT_SET_REGISTER_0
- I2C_I2C_CLKEN_OVERRIDE_0
- I2C_I2C_DEBUG_CONTROL_0
- I2C_I2C_CLK_DIVISOR_REGISTER_0
- I2C_I2C_CNFG_0
- I2C_I2C_CONFIG_LOAD_0

All other registers/fields have no meaning in packet mode and should be used only normal mode.

35.8.2 Programming Packet Header and Payload

Software should post the packets to I2C_TX_PACKET_FIFO. It writes the packet header followed by the payload to the I2C_TX_PACKET_FIFO register. The header size can vary from 3 to 4 words. For I2C, the size is 3 words header for request packets and 4 words for response packets. The first two words of the header contain a generic header. The third word of the packet (and fourth as well for response packets) contains I2C transaction-specific information. The payload contains the actual data to be written to the slave. In case of read operations, payload is nil, and hence, the packet contains only a header.

35.8.3 Reading from the RX FIFO

The data received from the I2C bus is pushed into the I2C_I2C_RX_FIFO_0. In PIO mode, a request interrupt is generated when the FIFO attention level is reached and if the corresponding interrupt enable bit is set. Software then reads the I2C_I2C_RX_FIFO_0 register to get the data. In DMA mode, a request is asserted to DMA after the attention level is reached.

35.8.4 Programming REPEAT_START/STOP bit in the protocol specific header

This field indicates whether to put a stop or repeated-start condition after the current transaction. By default, this bit is zero and a stop condition is put on the bus. If this bit is set to 1, a repeat start is put on the bus instead before proceeding with the next packet. REPEAT_START/STOP bit can be used for combining read operations and write operations within a single transaction with repeat start, or to do transfers beyond the 4Kbyte limit. This bit is present in the protocol specific header.

35.8.5 Programming ContinueXfer bit in the protocol specific header

I2C supports transfer sizes beyond 4Kbytes without repeat start condition again by combining multiple packets. This is in addition to the ability of using repeat start condition to do continuous transfers over 4Kbyte limit. With ContinueXfer, the current transfer would continue without stop or repeat start condition.

35.8.6 Transfers with START Byte

Transfers that need a longer start procedure can use the START Byte feature. In this mode, START Byte (8‘b00000001) is transmitted after the START condition and a dummy acknowledge (NACK) would be received. And actual transfer would start

with a Repeated START condition then. For START byte transfers, set SEND_START_BYTE to 1 in the protocol specific header of the packet.

35.8.7 Transfers with Slaves having No-ACK capability

There could be slave devices which do not generate ACK upon reception of a byte. I2C Master has the capability to do the transfers with these devices too. This would be done by setting CONTINUE_ON_NACK bit field of the protocol specific header to 1. In this case, the I2C master ignores the NACK bit and continues the transfers.

35.8.8 HS_MODE transfers

HS_MODE transfers are supported in packet mode only. For HS transfers, write HS master code byte in HS_MASTER_ADDR field of the protocol specific header and set HS_MODE bit to 1. If HS_MODE bit is programmed to 0, all the transfers would be done in non-HS modes (Standard/Fm/Fm+) based on the I2C clocks frequency programming. In HS mode, Master code bytes are transmitted in (Standard/Fm/Fm+) speed mode and the data transfers occurs in HS mode. I2C_I2C_HS_INTERFACE_TIMING_0_0 and I2C_I2C_HS_INTERFACE_TIMING_1_0 registers are used for HS timing generation on the I2C bus.

35.8.9 FIFO Control

In DMA mode, I2C_FIFO_CONTROL_0 [TX_FIFO_TRIG] indicates the number of words that need to be empty in I2C_I2C_TX_PACKET_FIFO_0 for the DMA trigger to be asserted. I2C_FIFO_CONTROL_0[RX_FIFO_TRIG] indicates the number of words that need to be full in I2C_I2C_RX_FIFO_0 for the DMA trigger to be asserted.

In PIO mode, INTERRUPT_STATUS_REGISTER [TFIFO_DATA_REQ] is set if the I2C_I2C_TX_PACKET_FIFO_0 empty count is more than the value programmed in TX_FIFO_TRIG. Similarly, INTERRUPT_STATUS_REGISTER [RFIFO_DATA_REQ] is set if I2C_I2C_RX_FIFO_0 full count is more than the value programmed in RX_FIFO_TRIG. CPU will be interrupted if status bits are set and their corresponding INT_EN is set in the INTERRUPT_MASK_REGISTER. The depth of TX and RX FIFOs is 8 words.

For Transmit and Receive Data requests in DMA or PIO mode, the I2C_I2C_CNFG_0[PACKET_MODE_EN] bit needs to be set to 1 along with FIFO_TRIG level settings. Otherwise, requests are not generated.

35.8.10 FIFO Status

This register indicates the number of entries that are empty in the TX FIFO (TX_FIFO_EMPTY_CNT) and the number of entries that are full in the RX FIFO (RX_FIFO_FULL_CNT).

35.8.11 Interrupt Status Register

This register (INTERRUPT_STATUS_REGISTER) gives the status of I2C Master in Packet Mode operation and also the status of Slave controller in FIFO mode and SMBUS timeout status. The register bits are set to 1 when the corresponding event happens. And the bit fields are sticky. Once set to 1, they will be stay 1. Write 1 to clear them to 0. However, TFIFO_DATA_REQ and RFIFO_DATA_REQ fields are exception to this as they depend on the FIFO trigger levels and cannot be cleared.

35.8.12 Interrupt Mask Register

This register (I2C_INTERRUPT_MASK_REGISTER_0) contains the interrupt enable bits for the status bits given in INTERRUPT_STATUS_REGISTER. A separate interrupt enable bit is available for each bit field of INTERRUPT_STATUS_REGISTER. Setting an INT_EN bit in this register enables signaling an interrupt to the system when a corresponding status bit is set in the INTERRUPT_STATUS_REGISTER. Clearing it masks the interrupt.

Note: *Interrupt enable bit fields “ARB_LOST_INT_EN/NOACK_INT_EN/TFIFO_OVF_INT_EN / RFIFO_UNF_INT_EN” not only just work as enables for interrupts generation of corresponding events in packet mode but also have special impact how the packets are parsed in packet*

mode. It is important to set them in packet mode. Refer to Section 35.8.15: Error Handling for details.

35.8.13 Interrupt Source Register

This is a read-only register (I2C_INTERRUPT_SOURCE_REGISTER) which returns the AND of Interrupt Mask and Interrupt Status Registers bit fields. A bit field in this register is set if the corresponding bit fields in I2C_INTERRUPT_MASK_REGISTER_0 and INTERRUPT_STATUS_REGISTER are set to 1.

Example:

I2C_INTERRUPT_SOURCE_REGISTER[ARB_LOST] = AND(I2C_INTERRUPT_MASK_REGISTER [ARB_LOST_INT_EN], INTERRUPT_STATUS_REGISTER [ARB_LOST])

35.8.14 Packet Transfer Status

In addition to I2C_INTERRUPT_STATUS_REGISTER_0, packet transfer status can be obtained from the I2C_PACKET_TRANSFER_STATUS_0 register as well as with additional information of the packet. Software can read this register to know which packet is currently being transferred, how many bytes are transferred in the on-going packet or packet is successfully transferred, or any Bus loss or NACK happened during the transfer.

1. TRANSFER_PKT_ID gives the ID of on-going packet or last transmitted packet ID.
2. TRANSFER_BYTENUM gives the information about the number of bytes that have been transferred so far in the current packet. Once the packet transfer is done, this field becomes zero.
3. TRANSFER_COMPLETE field is set when a packet (with STOP=1) is successfully transferred.
4. ARB_LOST shows the loss of the bus.
5. NOACK_FOR_ADDR for the Noack received for address byte.
6. NOACK_FOR_DATA for the Noack received for data byte.
7. CONTROLLER_BUSY shows PACKET_MODE_EN bit status. It would be 1 as long as packet mode is enabled.

35.8.15 Error Handling

In case of an error, for example, due to NOACK or loss of arbitration or TX-FIFO overflow or RX-FIFO underflow, the following steps describe the sequence to be followed before proceeding to any further transactions.

When an error occurs, the I2C controller stops further packet transfers and sets the corresponding status bits in I2C_INTERRUPT_STATUS_REGISTER for the event (ARB_LOST/NOACK/TFIFO_OVF/ RFIFO_UNF) to occur.

- PACKET_TRANSFER_STATUS register is updated with the current error packet ID, at which the error occurs.
- Software should reset the I2C controller.
- In the case of DMA transfer, DMA needs to be disabled.
- Release the reset to I2C controller.
- Redo the I2C configuration and restart the DMA if used.

In packet mode, special care should be taken to handle error recovery. If an error occurs during the transfer of a packet then the controller should be reset and DMA needs to be restarted (if using DMA transfer) and the entire packet needs to be resent since there is no accurate way of knowing how many bytes made it to the I2C slave.

The Repeat-Start bit in packet mode can complicate the situation further. When back-to-back packets are sent to a slave with repeat start bit set, and an error occurs, then it is not known during which packet transfer the error occurred. Therefore, the entire stream of packets that were sent back-to-back using repeat start needs to be resent.

On the receive case, it is simpler since if an error occurs, and bytes for a packet have already been received, then that packet can be deemed complete. Only incomplete transfers need to be retried.

Note: *Note: Hardware needs interrupts to be enabled in the beginning before a packet is posted to its TX-FIFO. Otherwise, the I2C controller could potentially parse stale data in the TX-FIFO in error cases, as described in the steps below.*

Hardware functionality in packet mode:

1. When packet_mode_en is 1, I2C hardware parses the entries (packets) in TX-FIFO, provided there is no pending errors status (for example, I2C_INTERRUPT_MASK_REGISTER [ARB_LOST_INT_EN] && INTERRUPT_STATUS_REGISTER [ARB_LOST] should be 0) that software has to attend for previous transfers.
2. If enough entries (minimum, packet header) are available in the FIFO, hardware parses the packet and starts driving the I2C bus. It puts the Start condition and address byte on the bus when the packet header is available. If payload (data) is also available along with packet header in the FIFO, it continues driving it and finishes the transfer. Otherwise, it holds the I2C bus until payload is posted by software.
3. If the I2C controller sees ARB_LOST or NACK during its transfer on the bus, it stops the transfer immediately, sets ARB_LOST/NACK to 1 in I2C status register and alerts software by signaling an interrupt to the system.
4. Then it flushes the FIFO and goes to IDLE state. It starts again here.
5. It checks if there are any pending interrupts that software has to attend.
6. ((ARB_LOST status && ARB_LOST_INT_EN) == 1).
7. If the condition is not true, it goes ahead and checks if any entries are in the FIFO. If yes, for hardware, it is a new packet posted by software after attending the ARB_LOST (if any) of the previous transaction.

Consider this case where software does not set interrupt enable bits in the beginning of the I2C controller configuration. It could lead to hardware parsing the stale data in the TX-FIFO.

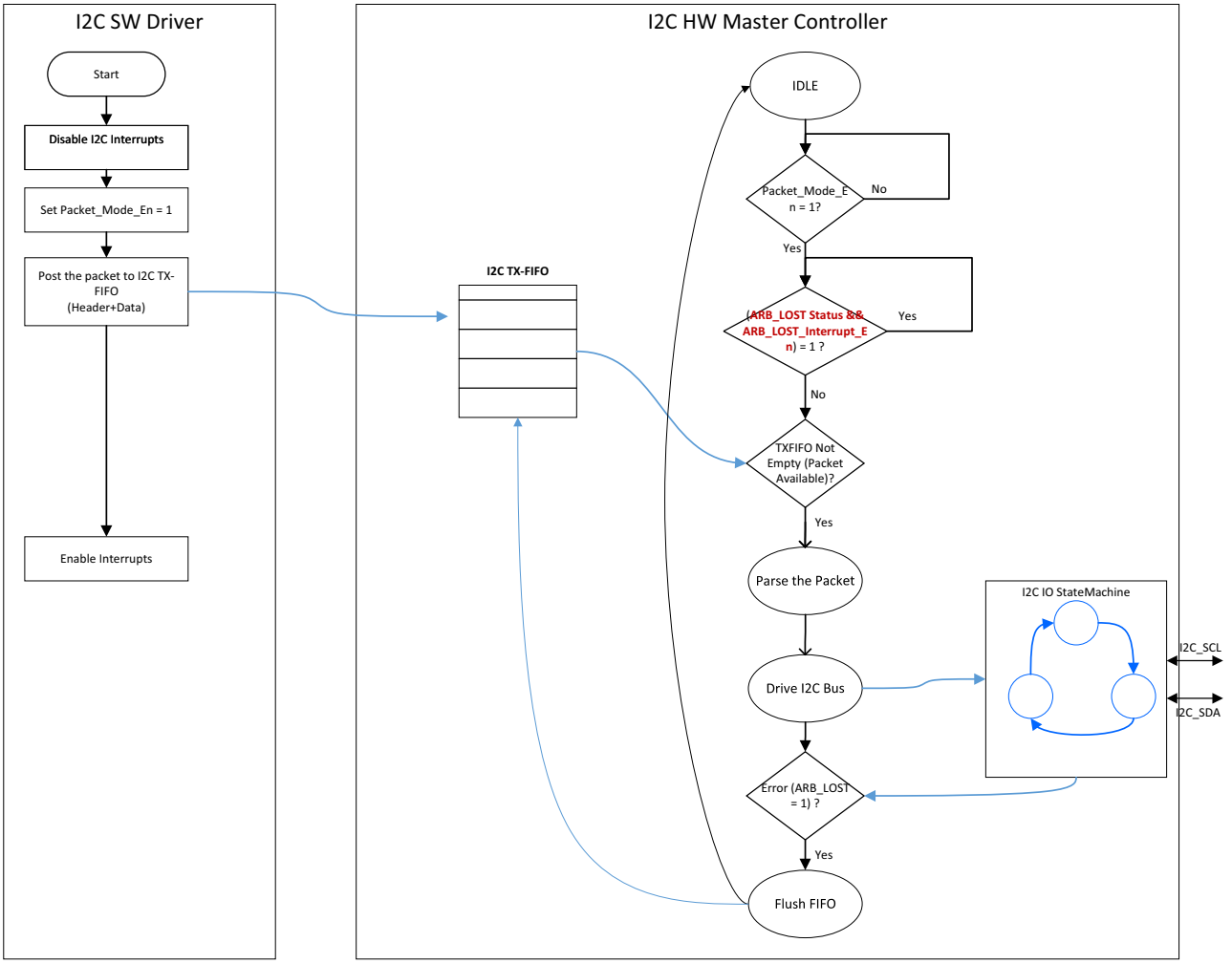
Example software I2C driver steps:

1. Disables all I2C interrupts in the beginning.
2. Sets packet_mode_en to 1.
3. Posts the packet to TX-FIFO. This triggers the hardware to initiate a transfer on the I2C bus when a minimum three entries (packet header) are available in the FIFO.
4. Fills the FIFO until it is full or the entire packet is posted if the packet size is less than or equal to FIFO size (32 bytes).
5. Enables the interrupts.
6. Wait on interrupts (transfer completion or error or tx_data_req to post remaining payload).

If hardware sees ARB_LOST from the I2C bus during software steps 3 and 4 above, this leaves a scope for hardware to parse the stale data in the FIFO as the interrupts are still disabled at this point. Software does not get interrupt so it continues to post the data and hardware, internally, finishes its FIFO flush step and comes to the idle state to check if hardware has posted any new packet. So there is an overlap here.

The above steps are described in the flowchart given below.

Figure 161: Software I2C Driver Flow



35.8.16 Slave Usage in FIFO Mode

Slave data transfers can be done using FIFO. Data can be transmitted and received through I2C_I2C_SLV_TX_PACKET_FIFO_0 and I2C_I2C_SLV_RX_FIFO_0 registers, respectively. For this, FIFO_XFER_EN bit in I2C_I2C_SL_CNFG_0 register needs to be set.

In FIFO-XFER mode, the slave by default works in one byte mode. For example, initially when a single byte is received, ACK is withheld and SLV_RX_BUFFER_FILLED interrupt is generated. Software should indicate whether to ACK this byte or not by writing into ACK_LAST_BYTE field of I2C_I2C_SL_CNFG_0 register. Note that ACK_LAST_BYTE_VALID field also needs to be set to indicate to hardware that ACK_LAST_BYTE is being updated. When software receives interrupt due to SLV_RX_BUFFER_FILLED, it can check ACK_WITHHELD of I2C_I2C_SLV_PACKET_STATUS_0 register to see if the slave is withholding the ACK for the last byte.

Software can update the buffer size from default to any number by programming I2C_I2C_SL_CNFG_0[BUFFER_SIZE].

35.8.17 WithHold Ack Feature

In FIFO mode, slave holds off ACK on the very last byte of transfer until the host explicitly indicates to I2C to release the bus by ACKing the last byte. Host is responsible for providing this indication to I2C slave in a timely fashion before timeouts happen in SMBUS applications.

35.8.18 Slave Read/Write Transfer Switching Status

Separate status bits are available to report slave data transfer switching from read to write and vice versa. I2C_INTERRUPT_STATUS_REGISTER_0 [SLV_RD2WR] indicates that there is a switch from read to write with repeated-start condition on the I2C bus and current Read transaction needs to be closed and start with a Write transaction. SLV_RD2WR status bit is set if the read to write switching happens before programmed buffer size completes. For example, the software programmed Slave buffer size to 10 bytes. But the Master device switched the transfer direction from current read to write before all 10bytes are transferred with a repeated-start on the bus. Then the SLV_RD2WR bit is set indicating read to write switching happened.

Similarly I2C_INTERRUPT_STATUS_REGISTER_0 [SLV_WR2RD] indicates the switching from Write to Read. Unlike SLV_RD2WR bit, this status bit is set if switching happens any time during transfer irrespective of the programmed buffer size is less than or equal to the actual data transferred.

35.9 Programming Guidelines for Master in Packet Mode

35.9.1 Programming Model for Master-Transmit Interrupt Method

1. Initial configuration

   Before accessing I2C controller, set up the following:

   a. I2C Clocks programming

   b. Pinmux programming

   c. I2C controller reset release

2. I2C Controller registers initialization:

   a. Write I2C_I2C_TLOW_SEXT_0 = 0x0.

   b. Write I2C_I2C_CMD_ADDR0_0 = 0x0.

   c. Write I2C_I2C_CMD_ADDR1_0 = 0x0.

   d. Write I2C_I2C_CMD_DATA1_0 = 0x0.

   e. Write I2C_I2C_CMD_DATA2_0 = 0x0.

   f. Write I2C_I2C_CLKEN_OVERRIDE_0 = 0x0.

   g. Write I2C_I2C_DEBUG_CONTROL_0 = 0x0.

   h. Write I2C_I2C_INTERRUPT_SET_REGISTER_0 = 0x0.

3. Program I2C_I2C_CLK_DIVISOR_REGISTER_0 register based on the required I2C bus speed. Refer to Section 35.6.1: I2C Frequency Divisor Register for details.

4. Configure I2C interface timing registers. Recommendation is to use default or PROD settings. But if needed, these registers can be programmed to achieve the required I2C interface timing:

   a. Write I2C_I2C_INTERFACE_TIMING_0_0 = 0x0204

   b. Write I2C_I2C_INTERFACE_TIMING_1_0 = 0x04070404

   c. Write I2C_I2C_HS_INTERFACE_TIMING_0_0 = 0x0308

   d. Write I2C_I2C_HS_INTERFACE_TIMING_1_0 = 0x0b0b0b0b

5. Configure FIFO trigger levels for data requests in PIO mode. The settings below are for example only. Program this bit field to the required trigger levels.

   a. Write I2C_FIFO_CONTROL_0[TX_FIFO_TRIG] = 0x3 (0x3 = System gets request from I2C controller when at least 4 words are empty in the I2C_I2C_TX_PACKET_FIFO_0)

6. Configure I2C Master to packet mode:

   a. Write I2C_I2C_CNFG_0[MULTI_MASTER_MODE] = DISABLE

b. Write I2C_I2C_CNFG_0[HS_RND_TRIP_DLY_EFFECT] = DISABLE.
c. Write I2C_I2C_CNFG_0[MSTR_CLR_BUS_ON_TIMEOUT] = 0.
d. Write I2C_I2C_CNFG_0[DEBOUNCE_CNT] = 0x2 //Example only, program as required. Refer to Section 35.6.1: I2C Frequency Divisor Register for details.
e. Write I2C_I2C_CNFG_0[NEW_MASTER_FSM] = ENABLE.
f. Write I2C_I2C_CNFG_0[PACKET_MODE_EN] = GO.
g. Write I2C_I2C_CNFG_0[SEND] = NOP.
h. Write I2C_I2C_CNFG_0[NOACK] = DISABLE.
i. Write I2C_I2C_CNFG_0[CMD2] = DISABLE.
j. Write I2C_I2C_CNFG_0[CMD1] = DISABLE.
k. Write I2C_I2C_CNFG_0[START] = DISABLE.
l. Write I2C_I2C_CNFG_0[SLV2] = DISABLE.
m. Write I2C_I2C_CNFG_0[LENGTH] = 0.
n. Write I2C_I2C_CNFG_0[A_MOD] = 0.

7. Activate the configuration:
   a. I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] = ENABLE
   b. I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] = DISABLE
   c. I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] = ENABLE
   d. Wait for I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] and I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] bits to get cleared by the hardware.
8. Enable Interrupts for I2C Master controller in packet mode:
   a. Write I2C_INTERRUPT_MASK_REGISTER_0[ARB_LOST_INT_EN] = ENABLE.
   b. Write I2C_INTERRUPT_MASK_REGISTER_0[NOACK_INT_EN] = ENABLE.
   c. Write I2C_INTERRUPT_MASK_REGISTER_0[TFIFO_OVF_INT_EN] = ENABLE.
   d. Write I2C_INTERRUPT_MASK_REGISTER_0[ALL_PACKETS_XFER_COMPLETE_INT_EN] = ENABLE.
   e. Write I2C_INTERRUPT_MASK_REGISTER_0[PACKET_XFER_COMPLETE_INT_EN] = ENABLE.
   f. Write I2C_INTERRUPT_MASK_REGISTER_0[TFIFO_DATA_REQ_INT_EN] = ENABLE.

This completes the configuration. Next step is, posting the packets to I2C controller to initiate data transfers on the I2C bus.

1. Packets posting based on TFIFO data requests:
   a. There could be one or more packets that need to be transferred from I2C.
   b. One packet transfer size (in words) = packet_header (in words) +payload_size (in words).
   c. Total transfer size = packet1_transfer_size + packet2_transfer_size +…….+packetN_transfer_size
   d. When the TX_FIFO attention level is reached, I2C sets the I2C_INTERRUPT_STATUS_REGISTER_0[TFIFO_DATA_REQ] bit and sends an interrupt to system if the interrupt is enabled with I2C_INTERRUPT_MASK_REGISTER_0[TFIFO_DATA_REQ_INT_EN] = ENABLE setting.
2. Transmit Data Request Interrupt handling:
   a. Read I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and check which bit fields are set to 1.
   b. For master transmit data requests, I2C_I2C_INTERRUPT_SOURCE_REGISTER_0[TFIFO_DATA_REQ]is set to 1.

c. Disable the interrupt by writing I2C_INTERRUPT_MASK_REGISTER_0[TFIFO_DATA_REQ_INT_EN] = DISABLE

d. Write the packet words to I2C_I2C_TX_PACKET_FIFO_0. The total words size that would be written for every tfifo_data_req request should be less than or equal to the FIFO trigger level. Otherwise, it will cause the TX_FIFO to overflow.

e. Decrement the total transfer size with the number of words written to the FIFO in the previous step (total_transfer_size = total_transfer_size - #words written to TX_FIFO). Check if the new transfer size is zero. If not zero, enable interrupt for TFIFO_DATA_REQ with I2C_INTERRUPT_MASK_REGISTER_0[TFIFO_DATA_REQ_INT_EN] = ENABLE and wait for the new interrupt.

The above steps continue until all the packets are posted to the I2C_I2C_TX_PACKET_FIFO_0.

Packet parsing: once a packet is available in the TX_FIFO, I2C parses it and starts doing transfers on the I2C bus. The transfers could be successful or unsuccessful. The following sections describe how to identify and handle such transfers.

1. Successful transfers:

   a. Once a packet is transmitted successfully by the I2C Master, I2C_INTERRUPT_STATUS_REGISTER_0[PACKET_XFER_COMPLETE] bit will be set to 1 to reflect the status. And if the packet header is configured to put STOP condition on the bus as the last packet of that transfer, I2C_INTERRUPT_STATUS_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] bit will also be set.

   b. If interrupt enables are set, corresponding bits are set in I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and signals an interrupt to the system.

2. Interrupt handling for successful transfers:

   a. Read I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and check which bit fields are set to 1.

   b. If a packet is transmitted successfully, I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 [PACKET_XFER_COMPLETE] is set. If the packet was configured to have STOP bit, I2C_I2C_INTERRUPT_SOURCE_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] would also be set.

   c. Clear them by writing-1 to I2C_INTERRUPT_STATUS_REGISTER_0[PACKET_XFER_COMPLETE] and I2C_INTERRUPT_STATUS_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] bits accordingly. This will clear the interrupt.

3. Transfers completion:

   a. Use PACKET_XFER_COMPLETE (if packets have RepeatedStart or ContinueXfer settings) or ALL_PACKETS_XFER_COMPLETE alerts (if packets have only STOP) to track how many packets are successfully transferred to the I2C bus. When all packets are transferred successfully to the bus, it matches the total alerts that I2C signals to the system.

1. Unsuccessful transfers:

   A packet transfer could be unsuccessful due to various reasons. I2C master logs this information in I2C_INTERRUPT_STATUS_REGISTER_0 register. A transfer could be unsuccessful due to I2C bus related ARB_LOST/NACK events or TX_PACKET_FIFO overflow errors.

   a. I2C_INTERRUPT_STATUS_REGISTER_0[ARB_LOST] is set to 1 if I2C master sees the bus is already occupied by some other device when a transfer is initiated or if a slave device pulls the SDA line low continuously for some unknown reason.

   b. I2C_INTERRUPT_STATUS_REGISTER_0[NOACK] is set to 1 if slave device responds with NACK for address or data byte transfer of a transaction.

   c. I2C_INTERRUPT_STATUS_REGISTER_0[TFIFO_OVF] is set to 1 if DMA or CPU writes to I2C_I2C_TX_PACKET_FIFO_0 even it is full.

   d. When any of these error events happens, I2C controller stops further packets transfers and flushes the data from the I2C_I2C_TX_PACKET_FIFO_0.

   e. If interrupt enables are set, corresponding bits are set in I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and signals an interrupt to the system.

2. Interrupt handling for unsuccessful transfers:

a. Read I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and check which bit fields are set to 1 (ARB_LOST/NOACK/TFIFO_OVF).

b. In all these cases, I2C controller reset is required. Follow the below sequence before going for further transfers

c. Reset the I2C controller.

d. Redo the Controller configuration. And go for the transfers.

Note: *I2C issues the DATA requests to the system based on FIFO trigger levels settings and the I2C_I2C_CNFG_0[PACKET_MODE_EN] setting. PACKET_MODE_EN bit needs to be set to 1 for request generation.*

35.9.2 Programming Model for Master-Transmit DMA Method

1. Initial configuration

   Before accessing I2C controller, set up the following:

   a. I2C Clocks programming

   b. Pinmux programming

   c. I2C controller reset release

2. I2C controller registers initialization:

   a. Write I2C_I2C_TLOW_SEXT_0 = 0x0.

   b. Write I2C_I2C_CMD_ADDR0_0 = 0x0.

   c. Write I2C_I2C_CMD_ADDR1_0 = 0x0.

   d. Write I2C_I2C_CMD_DATA1_0 = 0x0.

   e. Write I2C_I2C_CMD_DATA2_0 = 0x0.

   f. Write I2C_I2C_CLKEN_OVERRIDE_0 = 0x0.

   g. Write I2C_I2C_DEBUG_CONTROL_0 = 0x0.

   h. Write I2C_I2C_INTERRUPT_SET_REGISTER_0 = 0x0.

3. Program I2C_I2C_CLK_DIVISOR_REGISTER_0 register based on the required I2C bus speed. Refer to Section 35.6.1: I2C Frequency Divisor Register for details.

4. Configure I2C interface timing registers:

   Recommendation is to use default or PROD settings. But if needed, these registers can be programmed to achieve the required I2C interface timing.

   a. Write I2C_I2C_INTERFACE_TIMING_0_0 = 0x0204.

   b. Write I2C_I2C_INTERFACE_TIMING_1_0 = 0x04070404.

   c. Write I2C_I2C_HS_INTERFACE_TIMING_0_0 = 0x0308.

   d. Write I2C_I2C_HS_INTERFACE_TIMING_1_0 = 0x0b0b0b0b.

5. Configure FIFO trigger levels for DMA requests:

   The settings below are for example only. Program this bit field to the required trigger levels.

   a. Write I2C_FIFO_CONTROL_0[TX_FIFO_TRIG] = 0x3 (0x3 = DMA gets request from I2C controller when at least 4 words are empty in the I2C_I2C_TX_PACKET_FIFO_0).

6. Enable Interrupts for I2C Master controller in packet mode.

   a. Write I2C_INTERRUPT_MASK_REGISTER_0[ARB_LOST_INT_EN] = ENABLE.

   b. Write I2C_INTERRUPT_MASK_REGISTER_0[NOACK_INT_EN] = ENABLE.

c. Write I2C_INTERRUPT_MASK_REGISTER_0[TFIFO_OVF_INT_EN] = ENABLE.

d. Write I2C_INTERRUPT_MASK_REGISTER_0[ALL_PACKETS_XFER_COMPLETE_INT_EN] = ENABLE for tracking packet transfers completion status.

e. Write I2C_INTERRUPT_MASK_REGISTER_0[PACKET_XFER_COMPLETE_INT_EN] = ENABLE for tracking packet transfers completion status.

7. Post I2C packets (headers and data) to the memory for the write transfers. Refer to Section 35.5.1: Packet Flow for packets structure. Program RESP_PKT_ENABLE = 0 in the packet header.

8. Configure DMA for I2C transfers.

   Note the following when configuring DMA for I2C transfers:

   a. Set DMA source pointer to the memory location where the I2C packets are stored.

   b. Set DMA destination pointer to I2C instance baseaddress + I2C_I2C_TX_PACKET_FIFO offset.

   c. DMA burst size should match I2C_FIFO_CONTROL_0[TX_FIFO_TRIG] level settings.

   d. Configure DMA to Flow control mode.

   e. DMA transfer size should match the total packets size in words. E.g. 3 Write packets. 1st packet=3bytes transfer, 2nd packet=8bytes transfer and 3rd packet=11bytes transfer. Total size = 4+5+6=15 words.

9. Configure I2C Master to packet mode:

   a. Write I2C_I2C_CNFG_0[MULTI_MASTER_MODE] = DISABLE.

   b. Write I2C_I2C_CNFG_0[HS_RND_TRIP_DLY_EFFECT] = DISABLE.

   c. Write I2C_I2C_CNFG_0[MSTR_CLR_BUS_ON_TIMEOUT] = 0.

   d. Write I2C_I2C_CNFG_0[DEBOUNCE_CNT] = 0x2 //Example only, program as required. Refer to Section 35.6.1: I2C Frequency Divisor Register for details.

   e. Write I2C_I2C_CNFG_0[NEW_MASTER_FSM] = ENABLE.

   f. Write I2C_I2C_CNFG_0[PACKET_MODE_EN] = GO.

   g. Write I2C_I2C_CNFG_0[SEND] = NOP.

   h. Write I2C_I2C_CNFG_0[NOACK] = DISABLE.

   i. Write I2C_I2C_CNFG_0[CMD2] = DISABLE.

   j. Write I2C_I2C_CNFG_0[CMD1] = DISABLE.

   k. Write I2C_I2C_CNFG_0[START] = DISABLE.

   l. Write I2C_I2C_CNFG_0[SLV2] = DISABLE.

   m. Write I2C_I2C_CNFG_0[LENGTH] = 0.

   n. Write I2C_I2C_CNFG_0[A_MOD] = 0.

10. Activate the configuration:

    a. I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] = ENABLE.

    b. I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] = DISABLE.

    c. I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] = ENABLE.

    d. Wait for I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] and I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] bits to get cleared by the hardware.

    With this, it will initiate the write transfers from I2C master to the bus. The transfers could be successful or unsuccessful. The following sections describe how to identify and handle such transfers.

11. Successful transfers:

    a. Once a packet is transmitted successfully by the I2C Master, I2C_INTERRUPT_STATUS_REGISTER_0[PACKET_XFER_COMPLETE] bit will be set to 1 to reflect the status.

If the packet header configured to put STOP condition on the bus as the last packet of that transfer, I2C_INTERRUPT_STATUS_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] bit will also be set.

b. If interrupt enables are set, corresponding bits are set in I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and signals an interrupt to the system.

12. Interrupt handling for successful transfers:

    a. Read I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and check which bit fields are set to 1.

    b. If a packet is transmitted successfully, I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 [PACKET_XFER_COMPLETE] is set. And if the packet was configured to have STOP bit, I2C_I2C_INTERRUPT_SOURCE_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] would also be set.

    c. Clear them by writing-1 to I2C_INTERRUPT_STATUS_REGISTER_0[PACKET_XFER_COMPLETE] and I2C_INTERRUPT_STATUS_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] bits accordingly. This clears the interrupt.

13. Transfers completion:

    a. Check DMA status to know if all the packets are transferred to I2C controller from memory.

    b. Use PACKET_XFER_COMPLETE (if packets have RepeatedStart or ContinueXfer settings) or ALL_PACKETS_XFER_COMPLETE alerts (if packets have only STOP) to track how many packets are successfully transferred to the I2C bus. When all packets are transferred successfully to the bus, it matches the total number of alerts that I2C signals to the system.

1. Unsuccessful transfers

   A packet transfer could be unsuccessful due to various reasons. I2C master logs this information in I2C_INTERRUPT_STATUS_REGISTER_0 register. The transfer could be unsuccessful due to I2C bus related ARB_LOST/NACK events or TX_PACKET_FIFO Overflow errors.

   a. I2C_INTERRUPT_STATUS_REGISTER_0[ARB_LOST] will be set to 1 if I2C master sees the bus is already occupied by some other device when a transfer is initiated or if a slave device pulls the SDA line low continuously for some unknown reason.

   b. I2C_INTERRUPT_STATUS_REGISTER_0[NOACK] will be set to 1 if slave device responds with NACK for address or data byte transfer of a transaction.

   c. I2C_INTERRUPT_STATUS_REGISTER_0[TFIFO_OVF] will be set to 1 if DMA or CPU writes to I2C_I2C_TX_PACKET_FIFO_0 even it is full.

   d. When any of these error events happens, I2C controller stops further packets transfers and flushes the data from the I2C_I2C_TX_PACKET_FIFO_0.

   e. If interrupt enables are set, corresponding bits are set in I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and signals an interrupt to the system.

2. Interrupt handling for unsuccessful transfers:

   a. Read I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and check which bit fields are set to 1 (ARB_LOST/NOACK/TFIFO_OVF).

   b. In all these cases, I2C controller reset is required. Follow the sequence below before going for further transfers.

   c. Assert I2C controller reset.

   d. Disable DMA.

   e. Release I2C controller reset.

   f. Redo the I2C configuration, including DMA and go for the transfers.

Notes:

- *I2C issues the requests to DMA based on FIFO trigger levels settings and the I2C_I2C_CNFG_0[PACKET_MODE_EN] setting. PACKET_MODE_EN bit needs to be set to 1 for requests generation.*

- *Packet transfer status is also available in I2C_PACKET_TRANSFER_STATUS_0 register. SW can read this register to know which packet is currently being transferred (TRANSFER_PKT_ID), how many bytes are transferred in the on-going packet (TRANSFER_BYTENUM) or if the previous packet with STOP setting is successfully transferred (TRANSFER_COMPLETE and TRANSFER_PKT_ID) or any ARB_LOST/ NACK event happened (ARB_LOST, NOACK_FOR_ADDR/NOACK_FOR_DATA).*

35.9.3 Programming Model for Master-Receive DMA Method

1. Initial configuration

   Before accessing I2C controller, set up the following:

   a. I2C Clocks programming

   b. Pinmux programming

   c. I2C controller reset release

2. I2C controller registers initialization:

   a. Write I2C_I2C_TLOW_SEXT_0 = 0x0.

   b. Write I2C_I2C_CMD_ADDR0_0 = 0x0.

   c. Write I2C_I2C_CMD_ADDR1_0 = 0x0.

   d. Write I2C_I2C_CMD_DATA1_0 = 0x0.

   e. Write I2C_I2C_CMD_DATA2_0 = 0x0.

   f. Write I2C_I2C_CLKEN_OVERRIDE_0 = 0x0.

   g. Write I2C_I2C_DEBUG_CONTROL_0 = 0x0.

   h. Write I2C_I2C_INTERRUPT_SET_REGISTER_0 = 0x0.

3. Program I2C_I2C_CLK_DIVISOR_REGISTER_0 register based on the required I2C bus speed. Refer to Section 35.6.1: I2C Frequency Divisor Register for details.

4. Configure I2C interface timing registers.

   Recommendation is to use default or PROD settings. But if needed, these registers can be programmed to achieve the required I2C interface timing.

   a. Write I2C_I2C_INTERFACE_TIMING_0_0 = 0x0204.

   b. Write I2C_I2C_INTERFACE_TIMING_1_0 = 0x04070404.

   c. Write I2C_I2C_HS_INTERFACE_TIMING_0_0 = 0x0308.

   d. Write I2C_I2C_HS_INTERFACE_TIMING_1_0 = 0x0b0b0b0b.

5. Configure FIFO trigger levels for DMA requests.

   The settings below are for example only. Program the bit fields to the required trigger levels.

   a. Write I2C_FIFO_CONTROL_0[TX_FIFO_TRIG] = 0x1 (0x1 = DMA gets request from I2C controller when at least 2 words are empty in the I2C_I2C_TX_PACKET_FIFO_0).

   b. Write I2C_FIFO_CONTROL_0[RX_FIFO_TRIG] = 0x3 (0x3 = DMA gets request from I2C controller when at least 4 words are full in the I2C_I2C_RX_FIFO_0).

6. Enable Interrupts for I2C Master controller in packet mode:

   a. Write I2C_INTERRUPT_MASK_REGISTER_0[ARB_LOST_INT_EN] = ENABLE.

   b. Write I2C_INTERRUPT_MASK_REGISTER_0[NOACK_INT_EN] = ENABLE.

   c. Write I2C_INTERRUPT_MASK_REGISTER_0[TFIFO_OVF_INT_EN] = ENABLE.

   d. Write I2C_INTERRUPT_MASK_REGISTER_0[RFIFO_UNF_INT_EN] = ENABLE.

e. Write I2C_INTERRUPT_MASK_REGISTER_0[ALL_PACKETS_XFER_COMPLETE_INT_EN] = ENABLE for tracking packet transfers completion status.

f. Write I2C_INTERRUPT_MASK_REGISTER_0[PACKET_XFER_COMPLETE_INT_EN] = ENABLE for tracking packet transfers completion status.

7. Prepare and save I2C packets in the memory for the read transfers.

   Refer to Section 35.5.1: Packet Flow for packets structure. Program RESP_PKT_ENABLE = 0 in the packet header.

8. Configure DMA for I2C transfers.

   Only I2C programming details are given in this doc. DMA programming model is out of scope for this document. Refer to the DMA programming guidelines document. And note the following when configuring DMA for I2C transfers:

   a. For packets transfer from memory to I2C controller, if DMA CH0 is used.

   - Set DMA CH0 source pointer to the memory location where the I2C packets are stored.
   - Set DMA CH0 destination pointer to I2C instance base-address + I2C_I2C_TX_PACKET_FIFO offset.
   - DMA burst size should match I2C_FIFO_CONTROL_0[TX_FIFO_TRIG] level settings.
   - Configure DMA to Flow control mode.
   - DMA CH0 transfer size should match the total packets size in words. Since it is read packets only, only packet headers would be there. For example, 3 Read packets. Total size = 3+3+3= 9 words.

   b. To transfer the received-data from I2C controller to memory, if DMA CH1 is used.

   - Set DMA CH1 source pointer to I2C instance base-address + I2C_I2C_RX_FIFO_0 offset.
   - Set DMA CH1 destination pointer to the memory location where the I2C data needs to be saved.
   - DMA CH1 burst size should match I2C_FIFO_CONTROL_0[RX_FIFO_TRIG] level settings.
   - Configure DMA to Flow control mode.
   - DMA Transfer size should match the total words that need to be read from I2C_I2C_RX_FIFO_0.

9. Configure I2C Master to packet mode:

   a. Write I2C_I2C_CNFG_0[MULTI_MASTER_MODE] = DISABLE.

   b. Write I2C_I2C_CNFG_0[HS_RND_TRIP_DLY_EFFECT] = DISABLE.

   c. Write I2C_I2C_CNFG_0[MSTR_CLR_BUS_ON_TIMEOUT] = 0.

   d. Write I2C_I2C_CNFG_0[DEBOUNCE_CNT] = 0x2 //Example only, program as required. Refer to Section 35.6.1: I2C Frequency Divisor Register for details.

   e. Write I2C_I2C_CNFG_0[NEW_MASTER_FSM] = ENABLE.

   f. Write I2C_I2C_CNFG_0[PACKET_MODE_EN] = GO.

   g. Write I2C_I2C_CNFG_0[SEND] = NOP.

   h. Write I2C_I2C_CNFG_0[NOACK] = DISABLE.

   i. Write I2C_I2C_CNFG_0[CMD2] = DISABLE.

   j. Write I2C_I2C_CNFG_0[CMD1] = DISABLE.

   k. Write I2C_I2C_CNFG_0[START] = DISABLE.

   l. Write I2C_I2C_CNFG_0[SLV2] = DISABLE.

   m. Write I2C_I2C_CNFG_0[LENGTH] = 0.

   n. Write I2C_I2C_CNFG_0[A_MOD] = 0.

10. Activate the configuration

    a. I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] = ENABLE.

    b. I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] = DISABLE.

c. I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] = ENABLE.

d. Wait for I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] and I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] bits to get cleared by the hardware.

With this, it initiates the read transfers by the I2C master on the bus. The transfers could be successful or unsuccessful. The following sections describe how to identify and handle such transfers.

11. Successful transfers:

    a. Once a packet transfer is successfully done by the I2C Master, I2C_INTERRUPT_STATUS_REGISTER_0[PACKET_XFER_COMPLETE] bit is set to 1 to reflect the status. If the packet header configured to put STOP condition on the bus as the last packet of that transfer, I2C_INTERRUPT_STATUS_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] bit will also be set.

    b. If interrupt enables are set, corresponding bits are set in I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and signals an interrupt to the system.

12. Interrupt handling for successful transfers:

    a. Read I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and check which bit fields are set to 1.

    b. If a packet is transmitted successfully, I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 [PACKET_XFER_COMPLETE] is set. And if the packet was configured to have STOP bit, I2C_I2C_INTERRUPT_SOURCE_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] would also be set.

    c. Clear them by writing-1 to I2C_INTERRUPT_STATUS_REGISTER_0[PACKET_XFER_COMPLETE] and I2C_INTERRUPT_STATUS_REGISTER_0[ALL_PACKETS_XFER_COMPLETE] bits accordingly. This clears the interrupt.

13. Transfers completion:

    a. Check DMA status to know if all the packets are transferred to I2C controller from memory.

    b. Use PACKET_XFER_COMPLETE (if packets have RepeatedStart or ContinueXfer settings) or ALL_PACKETS_XFER_COMPLETE (if packets have only STOP) alerts to track how many packets are successfully transferred to the I2C bus. When all packets are transferred successfully to the bus, it matches the total number of alerts that I2C sends to the system.

    c. Check DMA status if it read all the data bytes from I2C controller and saved it to the memory.

1. Unsuccessful transfers:

   A packet transfer could be unsuccessful due to various reasons. I2C master logs this information in I2C_INTERRUPT_STATUS_REGISTER_0 register. The transfer could be unsuccessful due to I2C bus related ARB_LOST/NACK events or TX_PACKET_FIFO overflow errors or I2C_I2C_RX_FIFO_0 underflow errors.

   a. I2C_INTERRUPT_STATUS_REGISTER_0[ARB_LOST] will be set to 1 if I2C master sees the bus is already occupied by some other device when a transfer is initiated or if a slave device pulls the SDA line low continuously for some unknown reason.

   b. I2C_INTERRUPT_STATUS_REGISTER_0[NOACK] will be set to 1 if slave device responds with NACK for address or data byte transfer of a transaction.

   c. I2C_INTERRUPT_STATUS_REGISTER_0[TFIFO_OVF] will be set to 1 if DMA or CPU writes to I2C_I2C_TX_PACKET_FIFO_0 even it is full.

   d. I2C_INTERRUPT_STATUS_REGISTER_0[RFIFO_UNF] will be set to 1 if DMA or CPU reads I2C_I2C_RX_FIFO_0 even it is empty.

   e. When any of these error events happens, I2C controller stops further packets transfers and flushes the data from the FIFOs.

   f. If interrupt enables are set, corresponding bits are set in I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and signals an interrupt to the system.

2. Interrupt handling for unsuccessful transfers:

   a. Read I2C_I2C_INTERRUPT_SOURCE_REGISTER_0 register and check which bit fields are set to 1 (ARB_LOST/NOACK/TFIFO_OVF/ RFIFO_UNF).

b. In all these cases, I2C controller reset is required. Follow the sequence below before going for further transfers

c. Assert I2C controller reset.

d. Disable DMA.

e. Release I2C controller reset.

f. Redo the I2C configuration, including DMA. And go for the transfers.

35.10 Programming Guidelines for Slave in Byte Mode

35.10.1 Programming Model for Slave-Receive in 7-bit addressing Mode (transfer direction not changed)—Interrupt Method

1. Initial configuration

   Before accessing I2C controller, set up the following:

   a. I2C Clocks programming

   b. Pinmux programming

   c. I2C controller reset release

2. I2C controller registers initialization

   a. Write I2C_FIFO_CONTROL = 0x0.

   b. Write I2C_I2C_INTERRUPT_SET_REGISTER = 0x0.

   c. Write I2C_I2C_CLKEN_OVERRIDE = 0x0.

   d. Write I2C_I2C_DEBUG_CONTROL = 0x0.

   e. Write I2C_I2C_TLOW_SEXT_0 = 0x0.

   f. Write I2C_I2C_SL_INT_SET_0 = 0x0.

   g. Write I2C_I2C_SL_DELAY_COUNT_0 = 0x1e.

3. Select Slave Address Register and configure it in 7-bit addressing mode:

   a. Write I2C_I2C_SL_ADDR1_0[SL_ADDR1] = 0.

   b. Write 7-bit slave address in I2C_I2C_SL_ADDR1_0[SL_ADDR0] = {1'b0,SlaveAddress[6:0]}.

   c. To use SL_ADDR0 bit field for Slave address, write I2C_I2C_SL_ADDR2_0[SELECT_SLAVE] = 0.

   d. Write I2C_I2C_SL_ADDR2_0[SL1_ADDR_HI] = 0.

   e. Write I2C_I2C_SL_ADDR2_0[SL1_VLD] = SEVEN_BIT_ADDR_MODE.

   f. Write I2C_I2C_SL_ADDR2_0[SL_ADDR_HI] = 0.

   g. Write I2C_I2C_SL_ADDR2_0[VLD] = SEVEN_BIT_ADDR_MODE.

4. Configure Interrupt Mask registers to enable the interrupts

   a. Write I2C_INTERRUPT_MASK_REGISTER_0 = 0x0.

   b. Write I2C_I2C_SL_INT_MASK_0[HW_MSTR_INT] = DISABLE.

   c. Write I2C_I2C_SL_INT_MASK_0[REPROG_SL] = DISABLE.

   d. Write I2C_I2C_SL_INT_MASK_0[RST_SL] = DISABLE.

   e. Write I2C_I2C_SL_INT_MASK_0[END_TRANS] = ENABLE.

   f. Write I2C_I2C_SL_INT_MASK_0[SL_IRQ] = ENABLE.

   g. Write I2C_I2C_SL_INT_MASK_0[RCVD] = ENABLE.

   h. Write I2C_I2C_SL_INT_MASK_0[ZA] = DISABLE.

5. Configure SL_CNFG register to enable the Slave Controller:
   a. Write I2C_I2C_SL_CNFG_0[ENABLE_SL] = 1.
   b. Write I2C_I2C_SL_CNFG_0[PKT_MODE_EN] = 0.
   c. Write I2C_I2C_SL_CNFG_0[NEWSL] = 1.
   d. Write I2C_I2C_SL_CNFG_0[FIFO_XFER_EN] = 0.
   e. Write I2C_I2C_SL_CNFG_0[SLV_XFER_ERR_CLK_STRETCH_EN] = 0.
   f. Write I2C_I2C_SL_CNFG_0[BUFFER_SIZE] = 0.
   g. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE_VALID] = 0.
   h. Write I2C_I2C_SL_CNFG_0[RESP] = 0.
   i. Write I2C_I2C_SL_CNFG_0[ACK_WITHHOLD_EN] = 0.
   j. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE] = 0.
   k. Write I2C_I2C_SL_CNFG_0[NACK] = DISABLE.
6. Activate the configuration with I2C_CONFIG_LOAD register programming:
   a. I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] = ENABLE.
   b. I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] = ENABLE.
   c. I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] = DISABLE
7. Wait for the hardware to clear the I2C_CONFIG_LOAD register bit fields:
   a. Wait until I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] becomes zero.
   b. Wait until I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] becomes zero.

   With this, Slave controller would be ready for the data transfers. Start a Write transaction from the I2C Master device.
8. Slave Response to Address Phase of the Write Transaction

   When Slave receives address byte from the bus and if the address matches its programmed address, it does the following:
   a. Sets I2C_I2C_SL_STATUS_0[RNW] bit field with received address byte[0] bit which indicates transfer direction. 0 = WRITE, 1=READ.
   b. Saves the received address byte in I2C_I2C_SL_RCVD_0 register.
   c. Sets I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating that a new transaction is received.
   d. Generates an interrupt to the system if interrupt enables bits (I2C_I2C_SL_INT_MASK_0[RCVD], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are set.
   e. And holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared.
9. New transaction Interrupt handling:
   a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.
   b. Each bit field signifies an event occurrence.
   c. If a bit field is set to 1, it means, the current interrupt is because of the relevant event to this bit.
   d. In the new transaction received case, I2C_I2C_SL_INT_SOURCE_0[RCVD] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.
   e. Check I2C_I2C_SL_STATUS_0[RNW] bit status. For write transactions, I2C_I2C_SL_STATUS_0[RNW] = WRITE is set.
   f. Clear I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them.
   g. This clears the interrupt and releases the SCL line allowing the Master to go to next phase of the transaction.

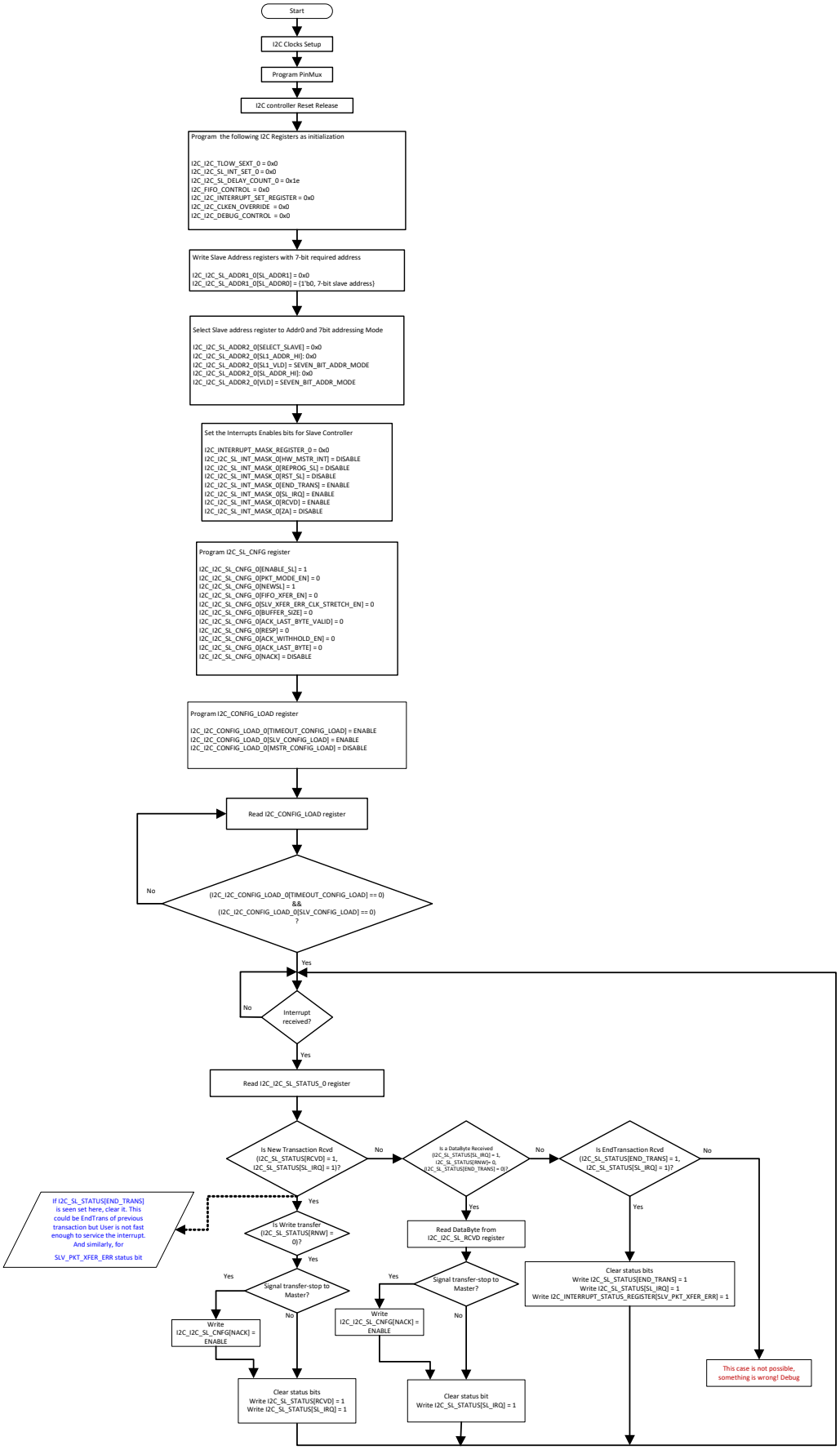
10. Slave Response to Data Phase of the Write Transaction
11. When a Master sends a data byte, the Slave controller does the following.
    a. Saves the received byte in I2C_I2C_SL_RCVD_0 register.
    b. Sets I2C_I2C_SL_STATUS_0[SL_IRQ] bit indicating a byte received.
    c. Generates an interrupt to the system if interrupt enable bit I2C_I2C_SL_INT_MASK_0[SL_IRQ] is set.
    d. Slave holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared.
12. Interrupts handling when received a data byte:
    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.
    b. In the data_byte received case, I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bit is set to 1
    c. Read the data byte from I2C_I2C_SL_RCVD_0 register
    d. To continue with the transfer, keep I2C_I2C_SL_CNFG_0[NACK] = DISABLE setting. Otherwise, to signal the Master to stop the transfer, configure I2C_I2C_SL_CNFG_0[NACK] = ENABLE
    e. Clear I2C_I2C_SL_STATUS_0[SL_IRQ] by writing-1 to it. This clears the interrupt and releases the SCL line allowing the Master to go ahead with next phase of the transaction
13. Slave Response to STOP condition of the Write transaction

    If Master sends STOP condition next, Slave does the following
    a. Sets I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating STOP condition is received
    b. Generates an interrupt to the system if Interrupt bits (I2C_I2C_SL_INT_MASK_0[END_TRANS], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are enabled.
14. Interrupts handling when Stop condition is received:
    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.
    b. In the STOP condition case, I2C_I2C_SL_INT_SOURCE_0[END_TRANS] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1
    c. Clear I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

Notes:

- *If Master terminates the transaction while the Slave controller is ready to accept the data, I2C_INTERRUPT_STATUS_REGISTER_0[SLV_PKT_XFER_ERR] bit is set at the end of the transaction when STOP condition is received. This bit can be cleared by writing-1 to it or it can be ignored.*
- *If the Slave controller sends NACK to any of the bytes during the write transfer, Master terminates the transaction after seeing NACK bit. In this case, No status bits (END_TRANS, SL_IRQ or SLV_PKT_XFER_ERR) is set when STOP condition is received. No interrupts either.*

Figure 162: Flowchart for Slave-Receive (transfer direction not changed) Programming Model



35.10.2 Programming Model for Slave-Transmit in 7-bit addressing Mode (transfer direction not changed)—Interrupt Method

1. Initial configuration

   Before accessing I2C controller, set up the following:

   a. I2C Clocks programming

   b. Pinmux programming

   c. I2C controller reset release

2. I2C controller registers initialization:

   a. Write I2C_FIFO_CONTROL = 0x0.

   b. Write I2C_I2C_INTERRUPT_SET_REGISTER = 0x0.

   c. Write I2C_I2C_CLKEN_OVERRIDE = 0x0.

   d. Write I2C_I2C_DEBUG_CONTROL = 0x0.

   e. Write I2C_I2C_TLOW_SEXT_0 = 0x0.

   f. Write I2C_I2C_SL_INT_SET_0 = 0x0.

   g. Write I2C_I2C_SL_DELAY_COUNT_0 = 0x1e.

3. Select Slave Address Register and configure it in 7-bit addressing mode:

   a. Write I2C_I2C_SL_ADDR1_0[SL_ADDR1] = 0.

   b. Write 7-bit slave address in I2C_I2C_SL_ADDR1_0[SL_ADDR0] = {1'b0, SlaveAddress[6:0]}.

   c. To use SL_ADDR0 bit field for Slave address, write I2C_I2C_SL_ADDR2_0[SELECT_SLAVE] = 0.

   d. Write I2C_I2C_SL_ADDR2_0[SL1_ADDR_HI] = 0.

   e. Write I2C_I2C_SL_ADDR2_0[SL1_VLD] = SEVEN_BIT_ADDR_MODE.

   f. Write I2C_I2C_SL_ADDR2_0[SL_ADDR_HI] = 0.

   g. Write I2C_I2C_SL_ADDR2_0[VLD] = SEVEN_BIT_ADDR_MODE.

4. Configure Interrupt Mask registers to enable the interrupts:

   a. Write I2C_INTERRUPT_MASK_REGISTER_0 = 0x0.

   b. Write I2C_I2C_SL_INT_MASK_0[HW_MSTR_INT] = DISABLE.

   c. Write I2C_I2C_SL_INT_MASK_0[REPROG_SL] = DISABLE.

   d. Write I2C_I2C_SL_INT_MASK_0[RST_SL] = DISABLE.

   e. Write I2C_I2C_SL_INT_MASK_0[END_TRANS] = ENABLE.

   f. Write I2C_I2C_SL_INT_MASK_0[SL_IRQ] = ENABLE.

   g. Write I2C_I2C_SL_INT_MASK_0[RCVD] = ENABLE.

   h. Write I2C_I2C_SL_INT_MASK_0[ZA] = DISABLE.

5. Configure SL_CNFG register to enable the Slave Controller:

   a. Write I2C_I2C_SL_CNFG_0[ENABLE_SL] = 1.

   b. Write I2C_I2C_SL_CNFG_0[PKT_MODE_EN] = 0.

   c. Write I2C_I2C_SL_CNFG_0[NEWSL] = 1.

   d. Write I2C_I2C_SL_CNFG_0[FIFO_XFER_EN] = 0.

   e. Write I2C_I2C_SL_CNFG_0[SLV_XFER_ERR_CLK_STRETCH_EN] = 0.

   f. Write I2C_I2C_SL_CNFG_0[BUFFER_SIZE] = 0.

g. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE_VALID] = 0.

h. Write I2C_I2C_SL_CNFG_0[RESP] = 0.

i. Write I2C_I2C_SL_CNFG_0[ACK_WITHHOLD_EN] = 0.

j. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE] = 0.

k. Write I2C_I2C_SL_CNFG_0[NACK] = DISABLE.

6. Activate the configuration with I2C_CONFIG_LOAD register programming:

   a. I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] = ENABLE.

   b. I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] = ENABLE.

   c. I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] = DISABLE.

7. Wait for the hardware to clear the I2C_CONFIG_LOAD register bit fields:

   a. Wait until I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] becomes zero.

   b. Wait until I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] becomes zero.

   With this, Slave controller would be ready for the data transfers. Start a Read transaction from the I2C Master device.

8. Slave Response to Address phase of Read transaction

   When Slave receives address byte from the bus and if the address matches its programmed address, it does the following.

   a. Sets I2C_I2C_SL_STATUS_0[RNW] bit field with received address byte[0] bit which indicates transfer direction. 0 = WRITE, 1=READ.

   b. Saves the received address byte in I2C_I2C_SL_RCVD_0 register.

   c. Sets I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating that a new transaction is received.

   d. Generates an interrupt to the system if Interrupt enables bits (I2C_I2C_SL_INT_MASK_0[RCVD], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are set.

   e. And holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared and a data byte is available in I2C_I2C_SL_RCVD_0 register.

9. New transaction Interrupt handling:

   a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

   b. In the new transaction received case, I2C_I2C_SL_INT_SOURCE_0[RCVD] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

   c. Check I2C_I2C_SL_STATUS_0[RNW] bit status. For read transactions, I2C_I2C_SL_STATUS_0[RNW] = READ is set.

   d. Clear I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

   e. Write Data byte to I2C_I2C_SL_RCVD_0 register. This releases the SCL line. The Master can read this byte now.

10. Slave Response to Data phase of Read transaction.

    If Master ACKs previous byte transfer, it means, Master wants to continue the read operation further. The Slave controller does the following in this case

    a. Sets I2C_I2C_SL_STATUS_0[SL_IRQ] bit indicating a request to transmit a data byte.

    b. Generates an interrupt to the system if interrupt enable bit I2C_I2C_SL_INT_MASK_0[SL_IRQ] is set.

    c. Slave holds the I2C bus by doing clock stretching (holds I2C SCL line low) until a data byte is available in I2C_I2C_SL_RCVD_0 register to transmit to the bus.

11. Interrupts handling in data byte transmit case:

    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

b. In the data transmit case, only I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bit is set to 1.

c. Clear I2C_I2C_SL_STATUS_0[SL_IRQ] bit by writing-1 to it. This clears the interrupt.

d. Write a Data byte to I2C_I2C_SL_RCVD_0 register. This will release the SCL line and allow the Master to continue the transfer.

12. Slave Response to End of the Read transaction

    If Master NACKs the previous byte transfer, it means, the Master wants to end the transaction. The Slave controller does the following then:

    a. Sets I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating the end of current transaction.

    b. Generates an interrupt to the system if interrupt bits (I2C_I2C_SL_INT_MASK_0[END_TRANS], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are enabled.

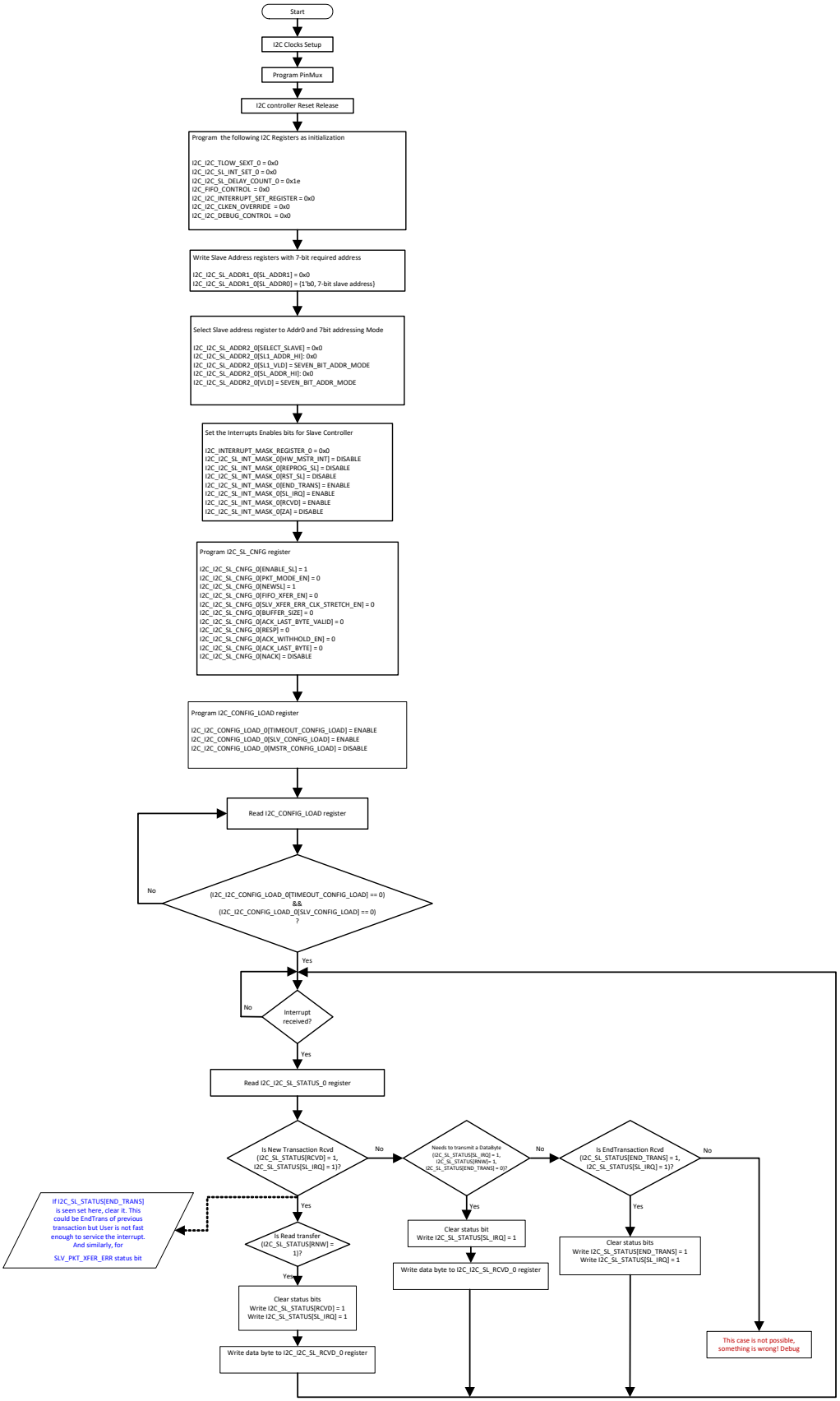
13. Interrupts handling when transaction END is received during read:

    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

    b. In the transaction END case, I2C_I2C_SL_INT_SOURCE_0[END_TRANS] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

    c. Clear I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

Figure 163: Flowchart for Slave-Transmit (transfer direction not changed) Programming Model



35.10.3 Programming Model for Slave-Receive-Transmit (transfer direction changed from Receive-to-Transmit with RepeatedStart) in 7-bit addressing Mode□Interrupt Method

1. Initial configuration

   Before accessing the I2C controller, set up the following.

   a. I2C Clocks programming

   b. Pinmux programming

   c. I2C controller reset release

2. I2C controller registers initialization:

   a. Write I2C_FIFO_CONTROL = 0x0.

   b. Write I2C_I2C_INTERRUPT_SET_REGISTER = 0x0.

   c. Write I2C_I2C_CLKEN_OVERRIDE = 0x0.

   d. Write I2C_I2C_DEBUG_CONTROL = 0x0.

   e. Write I2C_I2C_TLOW_SEXT_0 = 0x0.

   f. Write I2C_I2C_SL_INT_SET_0 = 0x0.

   g. Write I2C_I2C_SL_DELAY_COUNT_0 = 0x1e.

3. Select Slave Address Register and configure it in 7-bit addressing mode:

   a. Write I2C_I2C_SL_ADDR1_0[SL_ADDR1] = 0.

   b. Write 7-bit slave address in I2C_I2C_SL_ADDR1_0[SL_ADDR0] = {1'b0, SlaveAddress[6:0]}.

   c. To use SL_ADDR0 bit field for Slave address, write I2C_I2C_SL_ADDR2_0[SELECT_SLAVE] = 0.

   d. Write I2C_I2C_SL_ADDR2_0[SL1_ADDR_HI] = 0.

   e. Write I2C_I2C_SL_ADDR2_0[SL1_VLD] = SEVEN_BIT_ADDR_MODE.

   f. Write I2C_I2C_SL_ADDR2_0[SL_ADDR_HI] = 0.

   g. Write I2C_I2C_SL_ADDR2_0[VLD] = SEVEN_BIT_ADDR_MODE.

4. Configure Interrupt Mask registers to enable the interrupts:

   a. Write I2C_INTERRUPT_MASK_REGISTER_0[SLV_WR2RD_INT_EN] = ENABLE

   b. Write I2C_I2C_SL_INT_MASK_0[HW_MSTR_INT] = DISABLE.

   c. Write I2C_I2C_SL_INT_MASK_0[REPROG_SL] = DISABLE.

   d. Write I2C_I2C_SL_INT_MASK_0[RST_SL] = DISABLE.

   e. Write I2C_I2C_SL_INT_MASK_0[END_TRANS] = ENABLE.

   f. Write I2C_I2C_SL_INT_MASK_0[SL_IRQ] = ENABLE.

   g. Write I2C_I2C_SL_INT_MASK_0[RCVD] = ENABLE.

   h. Write I2C_I2C_SL_INT_MASK_0[ZA] = DISABLE.

5. Configure SL_CNFG register to enable the Slave Controller:

   a. Write I2C_I2C_SL_CNFG_0[ENABLE_SL] = 1.

   b. Write I2C_I2C_SL_CNFG_0[PKT_MODE_EN] = 0.

   c. Write I2C_I2C_SL_CNFG_0[NEWSL] = 1.

   d. Write I2C_I2C_SL_CNFG_0[FIFO_XFER_EN] = 0.

   e. Write I2C_I2C_SL_CNFG_0[SLV_XFER_ERR_CLK_STRETCH_EN] = 0.

   f. Write I2C_I2C_SL_CNFG_0[BUFFER_SIZE] = 0.

g. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE_VALID] = 0.

h. Write I2C_I2C_SL_CNFG_0[RESP] = 0.

i. Write I2C_I2C_SL_CNFG_0[ACK_WITHHOLD_EN] = 0.

j. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE] = 0.

k. Write I2C_I2C_SL_CNFG_0[NACK] = DISABLE.

6. Activate the configuration with I2C_CONFIG_LOAD register programming:

   a. I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] = ENABLE

   b. I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] = ENABLE

   c. I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] = DISABLE

7. Wait for the hardware to clear the I2C_CONFIG_LOAD register bit fields:

   a. Wait until I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] becomes zero.

   b. Wait until I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] becomes zero.

   With this, Slave controller would be ready for the data transfers. Start a transaction from the I2C Master device.

8. Slave Response to Address phase of Write transaction:

   When Slave receives address byte from the bus and if the address matches its programmed address, it does the following:

   a. Sets I2C_I2C_SL_STATUS_0[RNW] bit field with received address byte[0] bit which indicates transfer direction. 0 = WRITE, 1=READ.

   b. Saves the received address byte in I2C_I2C_SL_RCVD_0 register.

   c. Sets I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating that a new transaction is received.

   d. Generates an interrupt to the system if interrupt enables bits (I2C_I2C_SL_INT_MASK_0[RCVD], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are set.

   e. And holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared.

9. New transaction Interrupt handling:

   a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

   b. Each bit field signifies an event occurrence.

   c. If a bit field is set to 1, it means, the current interrupt is because of the relevant event to this bit.

   d. In the new transaction received case, I2C_I2C_SL_INT_SOURCE_0[RCVD] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

   e. Check I2C_I2C_SL_STATUS_0[RNW] bit status. For write transactions, I2C_I2C_SL_STATUS_0[RNW] = WRITE is set.

   f. Clear I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them.

   g. This clears the interrupt and releases the SCL line allowing the Master to go to next phase of the transaction.

10. Slave Response to Data phase of the Write transaction

    When a Master sends a data byte, the Slave controller does the following:

    a. Saves the received byte in I2C_I2C_SL_RCVD_0 register.

    b. Sets I2C_I2C_SL_STATUS_0[SL_IRQ] bit indicating a byte received.

    c. Generates an interrupt to the system if interrupt enable bit I2C_I2C_SL_INT_MASK_0[SL_IRQ] is set.

    d. Slave holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared.

11. Interrupts handling when received a data byte:

a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

b. In the data_byte received case, I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bit is set to 1.

c. Read the data byte from I2C_I2C_SL_RCVD_0 register.

d. To continue with the transfer, keep I2C_I2C_SL_CNFG_0[NACK] = DISABLE setting. Otherwise, to signal the Master to stop the transfer, configure I2C_I2C_SL_CNFG_0[NACK] = ENABLE.

e. Clear I2C_I2C_SL_STATUS_0[SL_IRQ] by writing-1 to it. This clears the interrupt and releases the SCL line allowing the Master to go ahead with next phase of the transaction.

12. Slave Response to Write-to-Read Transfer Direction change with RepeatedStart condition

When a Master sends RepeatedStart condition and changes the transfer direction from write to read, the Slave controller does the following:

a. Sets I2C_I2C_SL_STATUS_0[RNW] bit field with received address byte[0] bit which indicates the new transfer direction. 0 = WRITE, 1=READ.

b. Saves the received address byte in I2C_I2C_SL_RCVD_0 register.

c. Sets I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating that a new transaction is received.

d. Generates an interrupt to the system if interrupt enables bits (I2C_I2C_SL_INT_MASK_0[RCVD], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are set.

e. And holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared and a data byte is available in I2C_I2C_SL_RCVD_0 register to transmit to the Master.

13. New transaction Interrupt handling:

a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

b. In the new transaction received case, I2C_I2C_SL_INT_SOURCE_0[RCVD] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

c. Check I2C_I2C_SL_STATUS_0[RNW] bit status. For read transactions, it would be I2C_I2C_SL_STATUS_0[RNW] = READ.

d. Clear I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

e. This will set I2C_INTERRUPT_STATUS_REGISTER_0[SLV_WR2RD] bit to 1 now. And interrupt will be generated. Clear the interrupt by writing-1 to I2C_INTERRUPT_STATUS_REGISTER_0[SLV_WR2RD] bit field

f. I2C_INTERRUPT_STATUS_REGISTER_0[SLV_TX_BUFFER_REQ] bit will be set now. This can be ignored.

g. Write Data byte to I2C_I2C_SL_RCVD_0 register. This will release the SCL line. The Master can read the byte now.

14. Slave Response to Data phase of Read transaction

If Master ACKs previous byte of the read transfer, it means, Master wants to continue the read operation further. The Slave controller does the following in this case:

a. Sets I2C_I2C_SL_STATUS_0[SL_IRQ] bit indicating a request to transmit a data byte.

b. Generates an interrupt to the system if interrupt enable bit I2C_I2C_SL_INT_MASK_0[SL_IRQ] is set.

c. Slave holds the I2C bus by doing clock stretching (holds I2C SCL line low) until a data byte is available in I2C_I2C_SL_RCVD_0 register to transmit to the bus.

15. Interrupts handling in data byte transmit case:

a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

b. In the data transmit case, only I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bit is set to 1.

c. Clear I2C_I2C_SL_STATUS_0[SL_IRQ] bit by writing-1 to it. This clears the interrupt.

d. Write a Data byte to I2C_I2C_SL_RCVD_0 register. This will release the SCL line and allow the Master to continue the transfer

16. Slave Response to End of the Read transaction

    If Master NACKs the previous byte transfer, it means, the Master wants to end the transaction. The Slave controller does the following then:

    a. Sets I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating the end of current transaction.

    b. Generates an interrupt to the system if interrupt bits (I2C_I2C_SL_INT_MASK_0[END_TRANS], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are enabled.

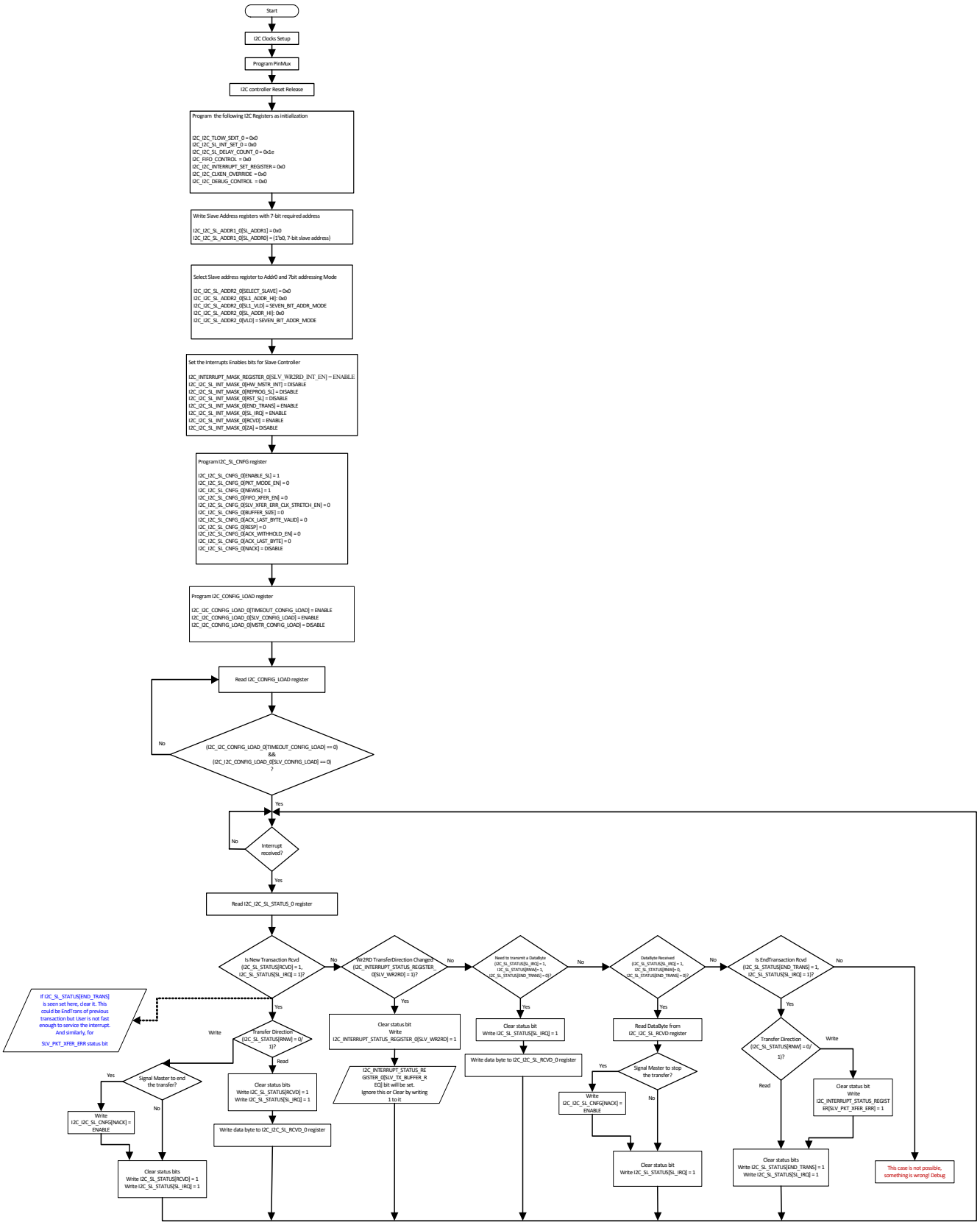
17. Interrupts handling when transaction END is received during read:

    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

    b. In the transaction END case, I2C_I2C_SL_INT_SOURCE_0[END_TRANS] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

    c. Clear I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

Figure 164: Flowchart for Slave-Receive-Transmit (transfer direction changed) Programming Model



35.10.4 Programming Model for Slave-Transmit-Receive (transfer direction changed from Transmit-to-Receive with RepeatedStart) in 7-bit addressing Mode□Interrupt Method

1. Initial configuration

   Before accessing the I2C controller, setup the following.

   a. I2C Clocks programming

   b. Pinmux programming

   c. I2C controller reset release

2. I2C controller registers initialization

   a. Write I2C_FIFO_CONTROL = 0x0

   b. Write I2C_I2C_INTERRUPT_SET_REGISTER = 0x0

   c. Write I2C_I2C_CLKEN_OVERRIDE = 0x0

   d. Write I2C_I2C_DEBUG_CONTROL = 0x0

   e. Write I2C_I2C_TLOW_SEXT_0 = 0x0

   f. Write I2C_I2C_SL_INT_SET_0 = 0x0

   g. Write I2C_I2C_SL_DELAY_COUNT_0 = 0x1e

3. Select Slave Address Register and configure it in 7-bit addressing mode:

   a. Write I2C_I2C_SL_ADDR1_0[SL_ADDR1] = 0

   b. Write 7-bit slave address in I2C_I2C_SL_ADDR1_0[SL_ADDR0] = {1'b0, SlaveAddress[6:0]}

   c. To use SL_ADDR0 bit field for Slave address, write I2C_I2C_SL_ADDR2_0[SELECT_SLAVE] = 0

   d. Write I2C_I2C_SL_ADDR2_0[SL1_ADDR_HI] = 0

   e. Write I2C_I2C_SL_ADDR2_0[SL1_VLD] = SEVEN_BIT_ADDR_MODE

   f. Write I2C_I2C_SL_ADDR2_0[SL_ADDR_HI] = 0

   g. Write I2C_I2C_SL_ADDR2_0[VLD] = SEVEN_BIT_ADDR_MODE

4. Configure Interrupt Mask registers to enable the interrupts

   a. Write I2C_INTERRUPT_MASK_REGISTER_0 = 0x0

   b. Write I2C_I2C_SL_INT_MASK_0[HW_MSTR_INT] = DISABLE

   c. Write I2C_I2C_SL_INT_MASK_0[REPROG_SL] = DISABLE

   d. Write I2C_I2C_SL_INT_MASK_0[RST_SL] = DISABLE

   e. Write I2C_I2C_SL_INT_MASK_0[END_TRANS] = ENABLE

   f. Write I2C_I2C_SL_INT_MASK_0[SL_IRQ] = ENABLE

   g. Write I2C_I2C_SL_INT_MASK_0[RCVD] = ENABLE

   h. Write I2C_I2C_SL_INT_MASK_0[ZA] = DISABLE

5. Configure SL_CNFG register to enable the Slave Controller:

   a. Write I2C_I2C_SL_CNFG_0[ENABLE_SL] = 1

   b. Write I2C_I2C_SL_CNFG_0[PKT_MODE_EN] = 0

   c. Write I2C_I2C_SL_CNFG_0[NEWSL] = 1

   d. Write I2C_I2C_SL_CNFG_0[FIFO_XFER_EN] = 0

   e. Write I2C_I2C_SL_CNFG_0[SLV_XFER_ERR_CLK_STRETCH_EN] = 0

   f. Write I2C_I2C_SL_CNFG_0[BUFFER_SIZE] = 0

g. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE_VALID] = 0

h. Write I2C_I2C_SL_CNFG_0[RESP] = 0

i. Write I2C_I2C_SL_CNFG_0[ACK_WITHHOLD_EN] = 0

j. Write I2C_I2C_SL_CNFG_0[ACK_LAST_BYTE] = 0

k. Write I2C_I2C_SL_CNFG_0[NACK] = DISABLE

6. Activate the configuration with I2C_CONFIG_LOAD register programming:

   a. I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] = ENABLE

   b. I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] = ENABLE

   c. I2C_I2C_CONFIG_LOAD_0[MSTR_CONFIG_LOAD] = DISABLE

7. Wait for the hardware to clear the I2C_CONFIG_LOAD register bit fields:

   a. Wait until I2C_I2C_CONFIG_LOAD_0[TIMEOUT_CONFIG_LOAD] becomes zero

   b. Wait until I2C_I2C_CONFIG_LOAD_0[SLV_CONFIG_LOAD] becomes zero

   With this, Slave controller would be ready for the data transfers. Start a transaction from I2C Master device.

8. Slave Response to Address phase of Read transaction

   When Slave receives address byte from the bus and if the address matches its programmed address, it does the following.

   a. Sets I2C_I2C_SL_STATUS_0[RNW] bit field with received address byte[0] bit which indicates transfer direction. 0 = WRITE, 1=READ

   b. Saves the received address byte in I2C_I2C_SL_RCVD_0 register

   c. Sets I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating that a new transaction is received.

   d. Generates an interrupt to the system if Interrupt enables bits (I2C_I2C_SL_INT_MASK_0[RCVD], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are set

   e. And holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared and a data byte is available in I2C_I2C_SL_RCVD_0 register

9. New transaction Interrupt handling:

   a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

   b. In the new transaction received case, I2C_I2C_SL_INT_SOURCE_0[RCVD] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1

   c. Check I2C_I2C_SL_STATUS_0[RNW] bit status. For read transactions, I2C_I2C_SL_STATUS_0[RNW] = READ is set.

   d. Clear I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

   e. Write Data byte to I2C_I2C_SL_RCVD_0 register. This will release the SCL line. The Master can read this byte now.

10. Slave Response to Data phase of Read transaction

    If Master ACKs previous byte transfer, it means, Master wants to continue the read operation further. The Slave controller does the following in this case.

    a. Sets I2C_I2C_SL_STATUS_0[SL_IRQ] bit indicating a request to transmit a data byte.

    b. Generates an interrupt to the system if interrupt enable bit I2C_I2C_SL_INT_MASK_0[SL_IRQ] is set

    c. Slave holds the I2C bus by doing clock stretching (holds I2C SCL line low) until a data byte is available in I2C_I2C_SL_RCVD_0 register to transmit to the bus.

11. Interrupts handling in data byte transmit case:

a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

b. In the data transmit case, only I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bit is set to 1

c. Clear I2C_I2C_SL_STATUS_0[SL_IRQ] bit by writing-1 to it. This clears the interrupt.

d. Write a Data byte to I2C_I2C_SL_RCVD_0 register. This will release the SCL line and allow the Master to continue the transfer.

12. Slave Response to Read-to-Write Transfer Direction change with RepeatedStart condition

    When Master wants to send RepeatedStart condition to change the transfer direction from read to write, it will NACK the last byte of the read transfer before sending Repeated condition.

    Slave does the following when this happens:

    a. Sets I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating the end of read transaction.

    b. Generates an interrupt to the system if interrupt enables bits (I2C_I2C_SL_INT_MASK_0[END_TRANS], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are set.

13. Interrupts handling when transaction END is received during read:

    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

    b. In the transaction END case, I2C_I2C_SL_INT_SOURCE_0[END_TRANS] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

    c. Clear I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

14. When New Transaction (READ) starts with RepeatedStart, Slave does the following after receiving the address byte of the new transaction:

    a. Sets I2C_I2C_SL_STATUS_0[RNW] bit field with received address byte[0] bit which indicates transfer direction. 0 = WRITE, 1=READ.

    b. Saves the received address byte in I2C_I2C_SL_RCVD_0 register.

    c. Sets I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating that a new transaction is received.

    d. Generates an interrupt to the system if interrupt enables bits (I2C_I2C_SL_INT_MASK_0[RCVD], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are set.

    e. And holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared.

15. New transaction Interrupt handling:

    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

    b. In the new transaction received case, I2C_I2C_SL_INT_SOURCE_0[RCVD] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

    c. Check I2C_I2C_SL_STATUS_0[RNW] bit status. For write transactions, I2C_I2C_SL_STATUS_0[RNW] = WRITE is set.

    d. Clear I2C_I2C_SL_STATUS_0[RCVD] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt and releases the SCL line allowing the Master to go to next phase of the transaction.

16. Slave Response to Data phase of the Write transaction:

    When a Master sends a data byte, the Slave controller does the following.

    a. Saves the received byte in I2C_I2C_SL_RCVD_0 register.

    b. Sets I2C_I2C_SL_STATUS_0[SL_IRQ] bit indicating a byte received.

    c. Generates an interrupt to the system if interrupt enable bit I2C_I2C_SL_INT_MASK_0[SL_IRQ] is set.

    d. Slave holds the I2C bus by doing clock stretching (holds I2C SCL line low) until the interrupt is cleared.

17. Interrupts handling when received a data byte:

a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

b. In the data_byte received case, I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bit is set to 1

c. Read the data byte from I2C_I2C_SL_RCVD_0 register.

d. To continue with the transfer, keep I2C_I2C_SL_CNFG_0[NACK] = DISABLE setting. Otherwise, to signal the Master to stop the transfer, configure I2C_I2C_SL_CNFG_0[NACK] = ENABLE.

e. Clear I2C_I2C_SL_STATUS_0[SL_IRQ] by writing-1 to it. This clears the interrupt and releases the SCL line allowing the Master to go ahead with next phase of the transaction.

18. Slave Response to STOP condition of the Write transaction

    If Master sends STOP condition next, Slave does the following:

    a. Sets I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits indicating STOP condition is received.

    b. Generates an interrupt to the system if Interrupt bits (I2C_I2C_SL_INT_MASK_0[END_TRANS], I2C_I2C_SL_INT_MASK_0[SL_IRQ]) are enabled.

19. Interrupts handling when Stop condition is received:

    a. Read I2C_I2C_SL_INT_SOURCE_0 register and check which bit fields are set to 1.

    b. In the STOP condition case, I2C_I2C_SL_INT_SOURCE_0[END_TRANS] and I2C_I2C_SL_INT_SOURCE_0[SL_IRQ] bits are set to 1.

    c. Clear I2C_I2C_SL_STATUS_0[END_TRANS] and I2C_I2C_SL_STATUS_0[SL_IRQ] bits by writing-1 to them. This clears the interrupt.

Notes:

- *If Master terminates the transaction while the Slave controller is ready to accept the data during write transfer, I2C_INTERRUPT_STATUS_REGISTER_0[SLV_PKT_XFER_ERR] bit is set at the end of the transaction when STOP condition is received. This bit can be cleared by writing-1 to it or it can be ignored.*
- *If the Slave controller sends NACK to any of the bytes during the write transfer, the Master terminates the transaction after seeing NACK bit. In this case, no status bits (END_TRANS, SL_IRQ or SLV_PKT_XFER_ERR) are set when the STOP condition is received. No interrupts either.*

35.11 I2C Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

35.11.1 I2C_I2C_CNFG_0

IC Controller Configuration Register (Master)

The I2C_CNFG register is used to configure:

- the number of bytes to be transmitted or received
- the slave device type (either a 7-bit device or a 10-bit device)
- enable mode to send Start-byte or not
- to select either a single slave transaction or two slave transaction
- enable mode to handle devices that do not generate an ACK.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00020800 (0bxxxxxxxxxxxxxxx100000100000000000)

Bit	Reset	Description
17	ENABLE	MULTI_MASTER_MODE: used to select single or multi master mode 0=single master 1=multi_master In single_master mode, no arbitration checks happen 0 = DISABLE 1 = ENABLE
15	0x0	MSTR_CLR_BUS_ON_TIMEOUT: When this bit is set, the I2C master will put STOP condition on the bus. 0 = NOP 1 = Put STOP condition on the I2C bus
14:12	0x0	DEBOUNCE_CNT: Debounce period for SDA and SCL lines. 0 = No debounce 1 = 2T 2 = 4T 3 = 6T, etc. where T is the period of the fixed PLL clock source coming to I2C. Maximum debounce period is 14T. A debounce period of > 50 ns is desirable.
11	0x1	NEW_MASTER_FSM: Maintained for backward compatibility. 0 = DISABLE 1 = ENABLE
10	0x0	PACKET_MODE_EN: Write 1 to initiate transfer in packet mode. 0 = NOP 1 = GO
9	0x0	SEND: Writing a 1 causes the master to initiate the transaction in normal mode. This bit is cleared by hardware when the transaction is done. Firmware should first configure all other registers and bits [8:0] of this register before this bit is programmed to one. 0 = NOP 1 = GO
8	0x0	NOACK: Enable mode to handle devices that do not generate an ACK. 1 = Do not look for an ACK at the end of the enable. 0 = DISABLE 1 = ENABLE
7	0x0	CMD2: Read/Write Command for Slave 2: 1 - Read Transaction; 0 - Write Transaction. For a 7-bit slave address, this bit must match with the LSB of address byte for slave 2. Valid only when bit 4 of this register is set.
6	0x0	CMD1: Read/Write Command for Slave 1: 1 - Read Transaction; 0 - Write Transaction. Command for Slave 1: For a 7-bit slave address, this bit must match with the LSB of address byte for slave 1.
5	0x0	START: 1 = Yes, a Start byte needs to be sent. 0 = DISABLE 1 = ENABLE
4	0x0	SLV2: 1 = Enables a two-slave transaction. 0 = No command for Slave 2 present.
3:1	0x0	LENGTH: The number of bytes to be transmitted per transaction. 000 = 1 byte ... 111 = 8 bytes. In a two slave transaction, the number of bytes should be programmed to be less than 011.
0	0x0	A_MOD: Address mode defines whether a 7-bit or a 10-bit slave address is programmed. 1 = 10-bit device address. 0 = 7-bit device address.

35.11.2 I2C_I2C_CMD_ADDR0_0

I2C Slave-1 Address

I2C_CMD_ADDR0 is programmed with the 7-bit or 10-bit address of slave 1 to which the transaction is intended.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR0: For 7-bit mode, the address is written in I2C_CMD_ADDR0[7:1], and I2C_CMD_ADDR0[0] indicates the read/write transaction. The I2C_CMD_ADDR0[0] bit must match the I2C_CNFG[6] bit. For 10-bit mode, the address is written in I2C_CMD_ADDR0[9:0], and I2C_CNFG[6] indicates the read/write transaction.

35.11.3 I2C_I2C_CMD_ADDR1_0

I2C Slave-2 Address

I2C_CMD_ADDR1 is programmed with the 7-bit or 10-bit address of slave 2 to which the transaction is intended.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR1: For 7-bit mode, the address is written in I2C_CMD_ADDR0[7:1], and I2C_CMD_ADDR0[0] indicates the read/write transaction. The I2C_CMD_ADDR0[0] bit must match the I2C_CNFG[7] bit. For 10-bit mode, the address is written in I2C_CMD_ADDR0[9:0], and I2C_CNFG[7] indicates the read/write transaction.

35.11.4 I2C_I2C_CMD_DATA1_0

I2C Controller Data 1: Transmit/Receive

The four least significant bytes of data to be transmitted are loaded into this register when the I2C Master is in Write mode.

The four least significant bytes of data are read through this register when the I2C Master is in Read mode.

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA4: Fourth data byte to be sent/received
23:16	0x0	DATA3: Third data byte to be sent/received
15:8	0x0	DATA2: Second data byte to be sent/received
7:0	0x0	DATA1: First data byte to be sent/received

35.11.5 I2C_I2C_CMD_DATA2_0

I2C Controller Data 2: Transmit/Receive

The four most significant bytes of data to be transmitted are loaded into the register when the I2C Master is in Write mode.

The four most significant bytes of data are read through this register when the I2C Master is in Read mode.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA8: Eighth data byte to be sent/received
23:16	0x0	DATA7: Seventh data byte to be sent/received
15:8	0x0	DATA6: Sixth data byte to be sent/received
7:0	0x0	DATA5: Fifth data byte to be sent/received

35.11.6 I2C_I2C_STATUS_0

I2C Controller Status (Master)

I2C_STATUS gives the status of the I2C Master operation.

Offset: 0x1c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	BUSY: 0 = NOT_BUSY 1 = BUSY
7:4	X	CMD2_STAT: Slave 2 status: Transaction for Slave 2 for byte x failed. x is 'h0 to 'ha. All other combinations are invalid. 0 = SL2_XFER_SUCCESSFUL 1 = SL2_NOACK_FOR_BYTE1 2 = SL2_NOACK_FOR_BYTE2 3 = SL2_NOACK_FOR_BYTE3 4 = SL2_NOACK_FOR_BYTE4 5 = SL2_NOACK_FOR_BYTE5 6 = SL2_NOACK_FOR_BYTE6 7 = SL2_NOACK_FOR_BYTE7 8 = SL2_NOACK_FOR_BYTE8 9 = SL2_NOACK_FOR_BYTE9 10 = SL2_NOACK_FOR_BYTE10
3:0	X	CMD1_STAT: Slave 1 status: Transaction for Slave 1 for byte x failed. x is 'h0 to 'ha. All other combinations are invalid. 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3 4 = SL1_NOACK_FOR_BYTE4 5 = SL1_NOACK_FOR_BYTE5 6 = SL1_NOACK_FOR_BYTE6 7 = SL1_NOACK_FOR_BYTE7 8 = SL1_NOACK_FOR_BYTE8 9 = SL1_NOACK_FOR_BYTE9 10 = SL1_NOACK_FOR_BYTE10

35.11.7 I2C_I2C_SL_CNFG_0

I2C Controller Configuration (Slave)

I2C_SL_CNFG register is used to configure, enable mode of slave ACK.

Enable mode of the slave response to the general call address.

The register should be programmed when the I^2C controller is configured as a slave.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxx00000000000000000100)

Bit	Reset	Description
20	0x0	FIFO_XFER_EN: If this bit is disabled, data is always communicated via the I2C_SL_RCVD register. If enabled, data is communicated through FIFOs 0 = DISABLE 1 = ENABLE
19:8	0x0	BUFFER_SIZE: Payload size in bytes
7	0x0	ACK_LAST_BYTE_VALID: Acknowledge the last byte valid (write-only). This bit qualifies the ACK_LAST_BYTE field. 0 = DISABLE 1 = ENABLE
6	0x0	ACK_LAST_BYTE: Acknowledge the last byte. 0 = DISABLE 1 = ENABLE
5	0x0	ACK_WITHHOLD_EN: ACK Withhold Feature Enable 0 = DISABLE 1 = ENABLE
4	0x0	PKT_MODE_EN: Packet Mode Enable 0 = DISABLE 1 = ENABLE
3	0x0	ENABLE_SL: By writing zero to this field, the slave can be turned off 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x1	NEWSL: New Slave. 1 - Use new slave. 0 = DISABLE 1 = ENABLE
1	0x0	NACK: Disable Slave ACK. 1 – The slave will not acknowledge reception of address or data byte. 0 = DISABLE 1 = ENABLE
0	0x0	RESP: Slave response to the general call address (zero address). 0 = DISABLE 1 = ENABLE

35.11.8 I2C_I2C_SL_RCVD_0

I2C Controller Slave Receive/Transmit Data (Slave)

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SL_DATA: Slave Received data

35.11.9 I2C_I2C_SL_STATUS_0

I2C Controller Slave Status (Slave)

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000XX0X (0bxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	R/W	Reset	Description
14:8	RO	X	HW_MSTR_ADR: Hardware master address received via general call addressing. This field is meaningful only if HW_MSTR_INT is set.
7	RW	0x0	HW_MSTR_INT: 1 = Interrupt has been generated by slave. Hardware Master Address is received after General Call Address. Received Hardware Master Address. 0 = No event.
6	RW	0x0	REPROG_SL: 1 = Interrupt has been generated by the slave after the General Call Address is 0x04. Reprogram slave address. 0 = No action.
5	RW	0x0	RST_SL: 1 = Interrupt has been generated by slave after the General Call Address is 0x06. Reset and reprogram slave address. 0 = No action.
4	RW	0x0	END_TRANS: 1 = Interrupt has been generated by slave. Transaction completed as indicated by stop/repeat start condition. 0 = No transaction occurred or transaction in progress.
3	RW	0x0	SL_IRQ: 1 = Interrupt has been generated by slave. 0 = No interrupt generated. 0 = UNSET 1 = SET
2	RW	0x0	RCVD: New Transaction Received status. 1 = Transaction occurred. 0 = No transaction occurred. 0 = NO_TRANSACTION_OCCURED 1 = TRANSACTION_OCCURED
1	RO	X	RNW: Slave Transaction status. 0 = WRITE 1 = READ
0	RW	0x0	ZA: Zero Address Status 1 = Yes, slave responded. 0 = No, slave did not respond. 0 = NO_SLAVE_RESPONSE 1 = SLAVE_RESPONSE

35.11.10 I2C_I2C_SL_ADDR1_0

I2C Controller Slave Address 1 Register (Slave)

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	SL_ADDR1: For a 10-bit slave address, this field is the least significant 8 bits.
7:0	0x0	SL_ADDR0: For a 10-bit slave address, this field is the least significant 8 bits.

35.11.11 I2C_I2C_SL_ADDR2_0

I2C Controller Slave Address 2 Register (Slave)

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxx000xxxxx000)

Bit	Reset	Description
16	0x0	SELECT_SLAVE: 0 = Use slave addr0. 1 = Use slave addr1.
10:9	0x0	SL1_ADDR_HI: In 7-bit address mode, these bits are don’t care. In 10-bit address mode, they represent the 2 MSBs of the address.
8	0x0	SL1_VLD: 0 = 7-bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE
2:1	0x0	SL_ADDR_HI: In 7-bit address mode, these bits are don’t care. In 10-bit address mode, they represent the 2 MSBs of the address.
0	0x0	VLD: 0 = 7-bit addressing. 1 - 10 bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE

35.11.12 I2C_I2C_TLOW_SEXT_0

I2C Controller SMBUS Timeout Thresholds

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27	0x0	RST_SL_ON_TIMEOUT: Reset Slave state machine on time-out
26	0x0	TLOW_MEXT_EN: Enable TLOW_MEXT counter
25	0x0	TLOW_SEXT_EN: Enable TLOW_SEXT counter
24	0x0	TIMEOUT_EN: Enable TIMEOUT counter
23:16	0x0	TLOW_MEXT: Cumulative clock low extend time (master device) accumulated over a byte transfer period in milliseconds (START to ACK, ACK to ACK, or ACK to STOP).
15:8	0x0	TLOW_SEXT: Cumulative clock low extend time (slave device) accumulated over a complete transfer (START till STOP)
7:0	0x0	TIMEOUT: Clock low time-out period in milliseconds

35.11.13 I2C_I2C_SL_DELAY_COUNT_0

I2C Slave Controller Delay Count

Offset: 0x3c | Read/Write: R/W | Reset: 0x0000001e (0bxxxxxxxxxxxxxxxx0000000000011110)

Bit	Reset	Description
15:0	0x1e	SL_DELAY_COUNT: The value determines the timing between an address cycle and a subsequent data cycle or two consecutive data cycles on the bus. The I2C_SL_DELAY_COUNT is valid only when an internal slave is accessed. I2C_SL_DELAY_COUNT has to be programmed such that TIMING = T * DLY, where T is the period of the clock source selected for I2C, DLY is I2C_SL_DELAY_COUNT, and TIMING is the desired timing. A value of >= 1250 ns is recommended.

35.11.14 I2C_I2C_SL_INT_MASK_0

I2C Controller Slave Mask (Slave)

Offset: 0x40 | Read/Write: R/W | Reset: 0x000000fd (0bxxxxxxxxxxxxxxxxxxxxxxxxx111111x1)

Bit	Reset	Description
7	0x1	HW_MSTR_INT: 0 = DISABLE 1 = ENABLE
6	0x1	REPROG_SL: 0 = DISABLE 1 = ENABLE
5	0x1	RST_SL: 0 = DISABLE 1 = ENABLE
4	0x1	END_TRANS: 0 = DISABLE 1 = ENABLE
3	0x1	SL_IRQ: 0 = DISABLE 1 = ENABLE
2	0x1	RCVD: 0 = DISABLE 1 = ENABLE
0	0x1	ZA: 0 = DISABLE 1 = ENABLE

35.11.15 I2C_I2C_SL_INT_SOURCE_0

I2C Controller Slave Source (Slave)

Offset: 0x44 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	HW_MSTR_INT: 0 = UNSET 1 = SET
6	X	REPROG_SL: 0 = UNSET 1 = SET
5	X	RST_SL: 0 = UNSET 1 = SET
4	X	END_TRANS: 0 = UNSET 1 = SET

Bit	Reset	Description
3	X	SL_IRQ: 0 = UNSET 1 = SET
2	X	RCVD: 0 = UNSET 1 = SET
0	X	ZA: 0 = UNSET 1 = SET

35.11.16 I2C_I2C_SL_INT_SET_0

I2C Controller Slave Source (Slave)

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	Reset	Description
7	0x0	HW_MSTR_INT: 0 = UNSET 1 = SET
6	0x0	REPROG_SL: 0 = UNSET 1 = SET
5	0x0	RST_SL: 0 = UNSET 1 = SET
4	0x0	END_TRANS: 0 = UNSET 1 = SET
3	0x0	SL_IRQ: 0 = UNSET 1 = SET
2	0x0	RCVD: 0 = UNSET 1 = SET
0	0x0	ZA: 0 = UNSET 1 = SET

Note: *Program the field PACKET_MODE_EN of I2C_CNFG register while working in packet mode.*

The set of registers given below describe the interface for packet mode only. With packet mode interface changes, normal mode registers and the operation are not affected. The transition from normal mode to packet mode is suggested because packet mode allows:

- There is no restriction on the number of bytes before and the after the repeated start
- The number of bytes that can be transferred with a single command is not limited to 8.

Though a packet can contain 4 Kbytes, because any number of packets can be pushed into the FIFO, there is no limit on the number of bytes that can be transferred.

- The transactions to different slaves can be chained together with repeat start and there is no limit on the number of slaves it can address.

35.11.17 I2C_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. For I2C, the header size is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction-specific information. The payload contains actual data to be written to the slave. For read operations, the payload is nil. Thus, the packet contains a header only.

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET: Software writes packets into this register. A packet may contain generic information

35.11.18 I2C_I2C_RX_FIFO_0

Header or I2C Specific Header or Data

Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: Software reads data from this register, causes a pop

35.11.19 I2C_PACKET_TRANSFER_STATUS_0

Offset: 0x58 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	TRANSFER_COMPLETE: The packet transfer for which the last packet is set has been completed 0 = UNSET 1 = SET
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet
3	X	NOACK_FOR_ADDR: No ACK received for the address byte 0 = UNSET 1 = SET
2	X	NOACK_FOR_DATA: No ACK received for the data byte 0 = UNSET 1 = SET
1	X	ARB_LOST: Arbitration lost for the current byte 0 = UNSET 1 = SET
0	X	CONTROLLER_BUSY:1 = Controller is busy 0 = UNSET 1 = SET

35.11.20 I2C_FIFO_CONTROL_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:13	0x0	SLV_TX_FIFO_TRIG: Slave Transmit FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is empty in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 words are empty in the FIFO
12:10	0x0	SLV_RX_FIFO_TRIG: Slave Receive FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
9	0x0	SLV_TX_FIFO_FLUSH:1 = Flush the TX FIFO. Cleared after the FIFO is flushed 0 = UNSET 1 = SET
8	0x0	SLV_RX_FIFO_FLUSH:1 = Flush the RX FIFO. Cleared after the FIFO is flushed 0 = UNSET 1 = SET
7:5	0x0	TX_FIFO_TRIG: Transmit FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is empty in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are empty in the FIFO

Bit	Reset	Description
4:2	0x0	RX_FIFO_TRIG: Receive FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
1	0x0	TX_FIFO_FLUSH:1 = Flush the TX FIFO. Cleared after the FIFO is flushed. 0 = UNSET 1 = SET
0	0x0	RX_FIFO_FLUSH:1 = Flush the RX FIFO. Cleared after the FIFO is flushed. 0 = UNSET 1 = SET

35.11.21 I2C_FIFO_STATUS_0

Offset: 0x60 | Read/Write: RO | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	SLV_XFER_ERR_REASON: This bit describes the nature of the packet transfer error. It is meaningful only if PKT_XFER_ERR is set. 0 = The Master terminated the transaction before it was completed. 1 = The Master did not terminate the transaction when all bytes are transferred.
23:20	X	SLV_TX_FIFO_EMPTY_CNT: The number of slots that can be written to the slave TX FIFO. 0000 = tx_fifo full. 0001 = 1 slot empty. 0010 = 2 slots empty
19:16	X	SLV_RX_FIFO_FULL_CNT: The number of slots to be read from the Slave RX FIFO. 0000 = rx_fifo empty. 0001 = 1 slot full. 0010 = 2 slots full
7:4	X	TX_FIFO_EMPTY_CNT: The number of slots that can be written to the TX FIFO. 0000 = tx_fifo full. 0001 = 1 slot empty. 0010 = 2 slots empty
3:0	X	RX_FIFO_FULL_CNT: The number of slots to be read from the RX FIFO. 0000 = rx_fifo empty. 0001 = 1 slot full. 0010 = 2 slots full

35.11.22 I2C_INTERRUPT_MASK_REGISTER_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xx00xxxx000000000000)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD_INT_EN: 0 = DISABLE 1 = ENABLE
27	0x0	SLV_RD2WR_INT_EN: 0 = DISABLE 1 = ENABLE
26	0x0	SLV_WR2RD_INT_EN: 0 = DISABLE 1 = ENABLE
25	0x0	SLV_PKT_XFER_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
24	0x0	SLV_TX_BUFFER_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
23	0x0	SLV_RX_BUFFER_FILLED_INT_EN: 0 = DISABLE 1 = ENABLE
22	0x0	SLV_PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21	0x0	SLV_TFIFO_OVF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
20	0x0	SLV_RFIFO_UNF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
17	0x0	SLV_TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
16	0x0	SLV_RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
11	0x0	BUS_CLEAR_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
10	0x0	TLOW_MEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
9	0x0	TLOW_SEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
8	0x0	TIMEOUT_INT_EN: 0 = DISABLE 1 = ENABLE
7	0x0	PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
5	0x0	TFIFO_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RFIFO_UNF_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	NOACK_INT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	ARB_LOST_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE

35.11.23 I2C_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set. If the interrupt is set, write a 1 to clear it. However, the TFIFO_DATA_REQ and RFIFO_DATA_REQ fields depend on the FIFO trigger levels and cannot be cleared.

Slv_rd2wr indicates there is a switch from read to write by repeat start and the current read transaction needs to be closed and started with a write transaction. Similarly, slv_wr2rd indicates a switch from write to read.

Offset: 0x68 | Read/Write: R/W | Reset: 0x000X000X (0bxxx000000000xxxxxxxx0000000000xx)

Bit	R/W	Reset	Description
28	RW	0x0	SLV_ACK_WITHHELD: ACK is withheld, waiting for software explicit information about the ACK 0 = UNSET 1 = SET
27	RW	0x0	SLV_RD2WR:Transaction switching from read to write 0 = UNSET 1 = SET
26	RW	0x0	SLV_WR2RD:Transaction switching from write to read 0 = UNSET 1 = SET
25	RW	0x0	SLV_PKT_XFER_ERR: 0 = Request was successful. 1 = Error has occurred during packet transfer 0 = UNSET 1 = SET
24	RW	0x0	SLV_TX_BUFFER_REQ: Slave TX buffer is empty 0 = UNSET 1 = SET
23	RW	0x0	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	RW	0x0	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET
21	RW	0x0	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	RW	0x0	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
17	RO	X	SLV_TFIFO_DATA_REQ: Slave TX FIFO data request 0 = UNSET 1 = SET
16	RO	X	SLV_RFIFO_DATA_REQ: Slave RX FIFO data request 0 = UNSET 1 = SET
11	RW	0x0	BUS_CLEAR_DONE: Bus clear done status 0 = UNSET 1 = SET
10	RW	0x0	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	RW	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	RW	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	RW	0x0	PACKET_XFER_COMPLETE: A packet has been transferred successfully. The TRANSFER_PKT_ID field can be used to know the current byte under transfer. This bit can be masked by the IE field in the I2C specific header. 0 = UNSET 1 = SET
6	RW	0x0	ALL_PACKETS_XFER_COMPLETE: All packets transferred successfully 0 = UNSET 1 = SET
5	RW	0x0	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
4	RW	0x0	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	RW	0x0	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	RW	0x0	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET
1	RO	X	TFIFO_DATA_REQ: TX FIFO data request 0 = UNSET 1 = SET
0	RO	X	RFIFO_DATA_REQ: RX FIFO data request 0 = UNSET 1 = SET

35.11.24 I2C_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- SCL frequency (Standard/Fast/FM+ modes) = ClkSourceFreq/((tlow+thigh+3)*(N+1)) for lower values of N, up to 3
- SCL frequency (Standard/Fast/FM+ modes) = ClkSourceFreq/((tlow+thigh+2)*(N+1)) for higher values of N, above 3
- SCL frequency (High Speed mode) = ClkSourceFreq/((ths_low+ths_high+4)*(N+1)) for lower values of N, up to 4
- SCL frequency (High Speed mode) = ClkSourceFreq/((ths_low+ths_high+2)*(N+1)) for higher values of N, above 4

Offset: 0x6c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000011001000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE: N = divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE: N = divide by n+1

35.11.25 I2C_I2C_INTERRUPT_SOURCE_REGISTER_0

This read-only register returns the AND of the Interrupt Source and Interrupt Mask registers.

Offset: 0x70 | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	X	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	X	SLV_RD2WR: 0 = UNSET 1 = SET
26	X	SLV_WR2RD: 0 = UNSET 1 = SET
25	X	SLV_PKT_XFER_ERR: Error occurred during a slave transfer 0 = UNSET 1 = SET
24	X	SLV_TX_BUFFER_REQ: Slave TX buffer is empty 0 = UNSET 1 = SET
23	X	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	X	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET

Bit	Reset	Description
21	X	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	X	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
17	X	SLV_TFIFO_DATA_REQ: Slave TX FIFO data request 0 = UNSET 1 = SET
16	X	SLV_RFIFO_DATA_REQ: Slave RX FIFO data request 0 = UNSET 1 = SET
11	X	BUS_CLEAR_DONE: Bus clear done 0 = UNSET 1 = SET
10	X	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	X	TLOW_SEXT_TIMEOUT: SMBUS sext time-out 0 = UNSET 1 = SET
8	X	TIMEOUT: SMBUS time-out 0 = UNSET 1 = SET
7	X	PACKET_XFER_COMPLETE: Packet transferred successfully 0 = UNSET 1 = SET
6	X	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	X	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	X	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	X	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	X	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET
1	X	TFIFO_DATA_REQ: TX FIFO data request 0 = UNSET 1 = SET
0	X	RFIFO_DATA_REQ: RX FIFO data request 0 = UNSET 1 = SET

35.11.26 I2C_I2C_INTERRUPT_SET_REGISTER_0

This write-only register can be used to set the interrupt status bit. A write to this register causes the bits in the status register to be set if the corresponding bit in the write data is 1'b1. A read always returns 'h0.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xxxxxxxx0000000000xx)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET

Bit	Reset	Description
27	0x0	SLV_RD2WR: 0 = UNSET 1 = SET
26	0x0	SLV_WR2RD: 0 = UNSET 1 = SET
25	0x0	SLV_PKT_XFER_ERR: Error occurred during slave transfer 0 = UNSET 1 = SET
24	0x0	SLV_TX_BUFFER_REQ: Slave TX buffer is empty 0 = UNSET 1 = SET
23	0x0	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	0x0	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET
21	0x0	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	0x0	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
11	0x0	BUS_CLEAR_DONE: Bus clear done 0 = UNSET 1 = SET
10	0x0	TLOW_MEXT_TIMEOUT:SMBUS mext time-out 0 = UNSET 1 = SET
9	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	0x0	PACKET_XFER_COMPLETE: Packet transferred successfully 0 = UNSET 1 = SET
6	0x0	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	0x0	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	0x0	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	0x0	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	0x0	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET

35.11.27 I2C_I2C_SLV_TX_PACKET_FIFO_0

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET: Software writes packets into this register. A packet may contain generic information.

35.11.28 I2C_I2C_SLV_RX_FIFO_0

Header or I2C Specific Header or Data

Offset: 0x7c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: Software reads data from this register, causes a pop.

35.11.29 I2C_I2C_SLV_PACKET_STATUS_0

Offset: 0x80 | Read/Write: RO | Reset: 0x0XXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	ACK_WITHHELD: Indicates that an ACK is withheld for the last byte and the slave is waiting for the host to explicitly command the slave to acknowledge the last byte. 0 = Bus is released. 1 = ACK is withheld
24	X	TRANSFER_COMPLETE: All the packets have been transferred successfully
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet

35.11.30 I2C_I2C_BUS_CLEAR_CONFIG_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00090004 (0bxxxxxxxx00001001xxxxxxxxxxxx100)

Bit	Reset	Description
23:16	0x9	BC_SCLK_THRESHOLD: send the clock pulses until this threshold is met
2	0x1	BC_STOP_COND: 0 = NO_STOP: Do not send a stop condition at the end of the bus clear operation 1 = STOP: Send a stop condition at the end of the bus clear operation
1	0x0	BC_TERMINATE: 0 = THRESHOLD: Irrespective of SDA release status during bus clear, terminate the bus clear only after the threshold is reached. 1 = IMMEDIATE: Terminate the bus clear operation immediately when SDA is released or threshold count is reached, whichever is earlier
0	0x0	BC_ENABLE: Starts bus clear operation. Hardware auto-clears this bit upon bus clear transaction completion

35.11.31 I2C_I2C_BUS_CLEAR_STATUS_0

Offset: 0x88 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	BC_STATUS: 0 = NOT_CLEARED: Indicates SDA is not released by the slave. Its status is still low. 1 = CLEARED: SDA is released

35.11.32 I2C_I2C_CONFIG_LOAD_0

Spare Register

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	TIMEOUT_CONFIG_LOAD: This bit loads the timeout configuration from the pclk domain to the receive (i2c_slow_clk) domain. Software has to set this bit finally after doing the required registers configuration for the logic to take the updates. Once the internal update is done, hardware auto clears this bit. Since the hardware would be busy with internal update, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	SLV_CONFIG_LOAD: This bit loads the slave configuration from the pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration for the slave controller to take updates. Once the internal update is done, hardware auto clears this bit. Since the hardware would be busy with internal update, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE
0	0x0	MSTR_CONFIG_LOAD: This bit loads the master configuration from the pclk domain to the receive (i2c_clk) domain. Software has to set this bit finally after doing the required registers configuration like I2C_I2C_CNFG_0 bit fields etc. Once the internal update is done, hardware auto clears this bit. Since the hardware would be busy with internal update, software should not write again until this bit is cleared by hardware. 0 = DISABLE 1 = ENABLE

35.11.33 I2C_I2C_INTERFACE_TIMING_0_0

Register for Standard/Fast/FM+ mode timing

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000204 (0bxxxxxxxxxxxxxxxxxx000010xx000100)

Bit	Reset	Description
13:8	0x2	THIGH: High period of the SCL clock
5:0	0x4	TLOW: Low period of the SCL clock

35.11.34 I2C_I2C_INTERFACE_TIMING_1_0

Register for Standard/Fast/FM+ mode timing

Offset: 0x98 | Read/Write: R/W | Reset: 0x04070404 (0bxx000100xx000111xx000100xx000100)

Bit	Reset	Description
29:24	0x4	TBUF: Bus free time between STOP and START conditions
21:16	0x7	TSU_STO: Setup time for STOP condition
13:8	0x4	THD_STA: Hold time for a (repeated) START condition
5:0	0x4	TSU_STA: Setup time for a Repeated START condition

35.11.35 I2C_I2C_HS_INTERFACE_TIMING_0_0

I2C interface timing register for HS mode transfers. Because HS master code is sent in Std/FM/FM+ modes, Standard/Fast/FM+ timing registers need to be programmed along with HS timing registers.

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000308 (0bxxxxxxxxxxxxxxxxxx000011xx001000)

Bit	Reset	Description
13:8	0x3	HS_THIGH: High period of the SCL clock
5:0	0x8	HS_TLOW: Low period of the SCL clock

35.11.36 I2C_I2C_HS_INTERFACE_TIMING_1_0

I2C interface timing register for HS mode transfers. Because HS master code is sent in Std/FM/FM+ modes, Standard/Fast/FM+ timing registers need to be programmed along with HS timing registers.

Offset: 0xa0 | Read/Write: R/W | Reset: 0x000b0b0b (0bxxxxxxxxxx001011xx001011xx001011)

Bit	Reset	Description
21:16	0xb	HS_TSU_STO: Setup time for STOP condition
13:8	0xb	HS_THD_STA: Hold time for a (repeated) START condition
5:0	0xb	HS_TSU_STA: Setup time for a Repeated START condition

CHAPTER 36: UART CONTROLLER

There are four Universal Asynchronous Receiver/Transmitters (UARTs) built into Tegra® X1 devices. These UARTs support both 16450 and 16550 compatible modes. A fifth UART is located in the Audio Processing Engine (APE).

36.1 Functional Description

36.1.1 UARTs A through D

All UARTs provide serial data synchronization and data conversion (parallel-to-serial and serial-to-parallel) for both receiver and transmitter sections. Synchronization for serial data stream is accomplished by adding start and stop bits to the transmit data to form a data character. Data integrity is accomplished by attaching a parity bit to the data character. The parity bit can be checked by the receiver for any transmission bit errors.

The interface supports word lengths from five to eight bits, an optional parity bit, and one or two stop bits. If enabled, parity can be odd, even, or forced to a defined state. Interrupts can be generated from any of 10 sources.

The UART controller supports both 16450 and 16550 compatible modes. The default mode is 16450. This mode provides independent 16-byte FIFOs for transmit and receive operations and is selected by the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit scratch register, 8 modem control lines, and 2 DMA handshake lines that are used to indicate when the FIFOs are ready to transfer data to the CPU.

The UARTs support a device clock of up to 200 MHz. Each symbol requires 16 clock cycles for proper sampling and processing of the input data stream. Thus, the maximum baud rate is 200/16 = 12.5M. Because 1 symbol = 1 bit, the data rate is 12.5 Mbps.

All four UARTs are identical.

36.1.2 Hardware Features

The features supported by UART A, UART B, UART C, and UART D are:

- Synchronization for the serial data stream with start and stop bits to transmit data to and from a data character
- Data integrity by attaching a parity bit to the data character
- Support for word lengths from 5 to 8 bits, with an optional parity bit, and 1 or 2 stop bits
- Support for both 16450- and 16550-compatible modes. The default mode is 16450.
- DMA capable for both TX and RX
- 8 bit x 36 deep TX FIFO
- 11 bit x 36 deep RX FIFO
  - 3 bits of 11 bits per entry will log the RX errors in FIFO mode (break, framing, and parity errors as bits 10,9,8 of the FIFO entry)
- Auto sense baud detection
- Timeout interrupts to indicate if the incoming stream stopped
- Flow control support on RTS and CTS
- Internal loopback
- SIR encoding/decoding (3/16 or 4/16 baud pulse widths to transmit bit 0)

36.1.3 Hardware Signaling

All UARTs in the Tegra X1 device are implemented by a hardware block that supports modem control signals. For all UARTs, the DSRn and DCDn input signals are tied high (active); the RIn input signal is tied low; and the DTRn output signal is not connected.

Table 228: Serial Bus Interface Signals

Signal Name	Description
Outputs	
TXD	Transmit Data Port
RTS	Request to Send
DTRn	Data Terminal Ready
Inputs	
RXD	Receiver Data Port
CTS	Clear to Send
DSRn	Data Set Ready
DCDn	Data Carrier Detect
RIn	Ring Indicator

Note: *The DSR, DCD and RI modem control signals do not control any hardware other than generating interrupts and status on change.*

36.2 UART Programming Guidelines

36.2.1 16450 Mode Programming

1. Program pin mux settings to select a UART
2. Enable the UART clocks
3. For enabling internal loopback, program MCR[4] to 1
4. Program FCR[0] to 0
5. Enable interrupts in the IER register as needed
6. Write data into the THR register
7. Wait for a THR interrupt, if enabled, or poll for LSR[5]
8. During a receive, wait for an RDR interrupt or poll for LSR[0]/LSR[9]
9. Read the UART.LSR register to clear interrupts

36.2.2 16550 Mode Programming

1. Program pin mux settings to select a UART
2. Enable the UART clocks
3. For enabling internal loopback, program MCR[4] to 1
4. Program FCR[0] to 1
5. Program trigger levels as required
6. Enable interrupts in the IER register as needed
7. Write data into the THR register
8. Wait for a THR interrupt, if enabled, or poll for LSR[5]

9. During a receive, wait for an RDR interrupt or poll for LSR[0]/LSR[9]
10. Read the UART.LSR register to clear interrupts
11. The APB-DMA requester numbers for UARTs A, B, C, and D are 8, 9, 10, and 19, respectively

36.2.3 Transmitter and Receiver Holding Registers

The serial transmitter section consists of a Transmit Hold Register (THR) and Transmit Shift Register (TSR). The status of transmit hold register is provided in the Line Status Register (LSR). Writing to this register (THR) transfers the contents of data bus (D [7:0]) to the Transmit Holding Register whenever the Transmit Holding Register or Transmit Shift Register is empty. The Transmit Holding Register empty flag is set to 1 when the transmitter is empty or data is transferred to the Transmit Shift Register. Note that a write operation is performed when the Transmit Holding Register empty flag is set.

The serial receiver section also contains an 8-bit Receiver Holding/Buffer Register (RBR). Receive data is removed from the UART and received by the processor by reading the RBR. The receiver contains a mechanism for preventing false starts as follows: On the falling edge of the start bit, the receiver internal counter starts to count 7.5 clocks (16x clock), which is the center of the start bit. If the input goes high before counter reaches mid-point sampling, RX_FSM is moved back to the IDLE state and that is treated as a start bit glitch. Verifying the start bit prevents the receiver from assembling a false data character due to a low going noise spike on the RX input. Receiver status codes are posted in the Line Status Register.

36.2.4 FIFO Interrupt Mode Operation

When the receive FIFO (FCR bit 0 = 1) and receive interrupts (IER bit 0 = 1) are enabled, a receiver interrupt occurs as follows:

- The receive data available interrupts are issued to the CPU when the FIFO has reached its programmed trigger level; it is cleared as soon as the FIFO drops below its programmed trigger level.
- The ISR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt, it is cleared when the FIFO drops below the trigger level.
- The data ready bit (LSR bit 0) is set as soon as a character is transferred from the shift register to the receiver FIFO. It is reset when the FIFO is empty.

36.2.5 FIFO Polled Mode Operation

When FCR bit 0 = 1; clearing IER bits [3:0] to zero puts the UART in the FIFO polled mode of operation. Since the receiver and transmitter are controlled separately, either one or both can be in the polled mode operation by using the line status register as follows:

- LSR bit 0 is set when the number of entries is greater than or equal to trigger level programmed
- LST bits [4:1] specify which error(s) have occurred
- LSR bit 5 indicates when the transmit FIFO is empty
- LSR bit 6 indicates when both transmit FIFO and transmit shift registers are empty
- LSR bit 7 indicates when there are any errors in the receive FIFO
- LSR bit 8 indicates when the transmit FIFO is full
- LSR bit 9 indicates RX FIFO empty status

The UART requires a two-step FIFO enable operation in order to enable receive trigger levels.

36.2.6 FIFO Enable Sequence Requirements

This sequence requires a minimum time interval of 3 * baud_clock_period between the point in time that the FIFO-enable write reaches the UART controller and the point in time that anything is written to the FIFO.

1. Write to the UART_FCR register to enable the FIFO.
2. Do a dummy read from a UART register with no read side-effects to ensure that the write reached the FIFO.

3. Delay for at least 3 * baud_clock_period before writing anything to the FIFO or expecting anything to enter the FIFO. (The delay is required to guarantee that FIFO is enabled internally in UART before any data is written to it.)

36.2.7 Programmable Baud Rate Generator

The UART contains a programmable baud rate generator that can take the UART clock input and divide it by any divisor from 1 to 2^{16}-1. Refer to the Clocking and Reset Controller section for definitions of the UART clocks. The output frequency of baudout is equal to 16X the transmission baud rate (Baudout=16 X baud rate). Customized baud rates are achieved by selecting proper divisor values for the MSB and LSB bits of the baud rate generator.

36.2.8 Enable Register (IER)

There is an Interrupt Enable Register for each UART. The Interrupt Enable Register(s) masks the incoming interrupts from the receiver ready, transmitter empty, line status, and modem status registers to the INT output pin.

36.2.9 Interrupt Identification Register (IIR)

The UART provides four level prioritized interrupt conditions to minimize software overhead during data character transfers. The Interrupt Identification Register (IIR) provides the source of the interrupt in a prioritized manner.

During the read cycle, the 16550 provides the highest interrupt level to be serviced by the CPU. No other interrupts are acknowledged until the particular interrupt is serviced. The prioritized interrupt levels are shown in the following table. The Receive Data Time-out mode is enabled when the UART is operating in the FIFO mode.

Receive time-out does not occur if receive FIFO is empty. The time-out counter is reset at the center of each stop bit received or each time the Receive Holding Register is read. The actual time out value is:

- T (Time out length in bits) = 4 X P (Programmed work length) + 12

To convert the time-out value to a character value, divide this number to its complete word length + parity (if used) + number of stop bits and start bit.

Example

If you program the word length = 7 with no parity and one stop bit, the time-out is:

- T = 4X7 (programmed word length) + 12 = 40 bits.
- Character time = 40/9
- [(Programmed word length = 7) + (stop bit = 1) + (start bit = 1)] =4.4 characters

Table 229: Prioritized Interrupt Levels

Priority	D3	D2	D1	D0	Interrupt Source Description
1	0	1	1	0	LSR (Receiver Line Status Register)
2	0	1	0	0	RXRDY (Received Data Ready)
2	1	1	0	0	RXRDY (Received Data Timeout)
3	0	0	1	0	TXRDY (Transmitter Holding Register)
4	0	0	0	0	MSR (Modem Status Register)

36.2.10 FIFO Control Register (FCR) Modes

The FIFO control register is used to enable the FIFOs, clear the FIFOs, set the receiver FIFO trigger level, and select the type of DMA signaling.

FCR bit 0 should be programmed to ‘1’ to operate the UART controller in FIFO (16450) mode. When FIFO mode is enabled, allow minimum of 3 baud cycles duration before writing TX_DATA into the THR_DLAB register.

The operation of the FCR in the four DMA modes is given below.

36.2.10.1 Transmit Operation in DMA Mode 0 or Mode 1

When the UART is in the 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 0) and when there are no characters in the transmit FIFO or Transmit Holding Register, the TXRDY* pin goes low. Once active, the TXRDY* pin goes high (inactive) after the first character is loaded into the Transmit Holding Register.

36.2.10.2 Receive Operation in DMA Mode 0

When the UART is in 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1,FCR bit 3 = 0) and there is at least 1 character in the receive FIFO, the RXRDY* pin goes low. Once active, the RXRDY* pin goes high (inactive) when there are no more characters in the receiver.

36.2.10.3 Receive Operation in DMA Mode 1

When the UART is in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 1) and the trigger level or the timeout has been reached, the RXRDY* pin goes low. Once it is activated, it goes high (inactive) when there are no more characters in the FIFO.

36.2.11 Line Control Register (LCR)

The Line Control Register is used to specify the asynchronous data communication format. The word length, stop bits, and parity can be selected by writing appropriate bits in this register.

36.2.12 Modem Control Register (MCR)

This register controls the interface with the modem or a peripheral device (RS232).

Loopback Mode

If MCR[4] = 1, the loopback mode is enabled, and the following occurs:

The receiver input (RX), CTS, DSR, CD, and RI are disabled. Internally, the transmitter output is connected to the receiver input and DTR, RTS, OP1, and OP2 are connected to modem control inputs.

In this mode, the receiver and transmitter interrupts are fully operational, but the interrupt sources are now the lower four bits of the Modem Control Register instead of the four modem control inputs. The interrupts are controlled by the IER register.

Table 230: Modem Control Register (MCR)

Bit	Name	Description
7	N/A	Not used. Set to 0 internally.
6	RTS_EN	1 = Enable Hardware Flow Control using RTS 0 = Disable Hardware Flow Control
5	CTS_EN	1 = Enable Hardware Flow Control using CTS 0 = Disable Hardware Flow Control
4	LOOP	0 = Normal operating mode. 1 = Enable local loop-back mode (diagnostic).
3	OUT2	nOUT2 polarity
2	OUT1	nOUT1 polarity
1	RTS	1 = Force RTS (Request to Send) low 0 = Force RTS to high
0	DTR	1 = Force DTR (Data Terminal Ready) to low 1 = Force to high

36.2.13 Line Status Register (LSR)

The Line Status Register provides the status of data transfer to the CPU.

36.2.14 Modem Status Register (MSR)

The modem status register provides the current state of the control lines from the modem or peripheral to the CPU. Four bits of this register are used to indicate the changed information. These bits are set to 1 whenever a control input from the modem changes state. They are set to 0 whenever the CPU reads the register.

36.2.15 Scratchpad Register (SR)

Eight bits of information can be stored in this register. Information in this register does not affect the operation of the device in any way.

36.2.16 Autobaud Sense Register (ASR)

The UART can automatically determine the correct baud divisor by using the Autobaud Sense Register (UART offsets 0x3C). The most significant bit of this register is the valid flag. A write to this register will clear the valid flag and enable the autobaud process. When the first RX edge occurs, a counter running at 24 MHz starts counting. When another rx_edge occurs, the "complete" flag is set, the value is frozen, and the Autobaud Sense Value register is updated with the count value. The low 20 bits of the ASR give the number of clocks within a single bit. Because the UART uses 16x oversampling, the resulting value needs to be adjusted by shifting right 4 bits, then loading the resulting count in the divisor latch of the UART. (In the code snippet below, the lower 4 bits are rounded to give slightly greater accuracy.)

Because the speed determination is made by measuring the start bit, special characters must be sent by the transmitting UART to guarantee that the next character after the start bit is a 1. Since bit 0 (rightmost bit) is sent first, the ASCII Carriage Return character (CR) is sufficient to enable proper speed sense.

The following code snippet returns the values for DLH and DLL in r9, r8:

```
MOV r0, #1  ; dummy write data
STRB    r0, [r3, #U_ASR]; Start autobaud sense (r3 as uart_base)
; now poll the autobaud sense register MSB until Valid is true
wait4valid
 LDR    r2, [r3, #U_ASR]; Read ASR
 TST    r2, #0x80000000 ; the Valid bit (active high)
 BEQ    wait4valid
; autobaud sense check complete...
; r2 as number of 1x clocks in one bit time
; representing the number of 24MHz clocks in the start bit
; Since this represents 16 of the baud (16x) clocks, we
; will be dividing by 16 to get baud divisor, but first
; round to nearest by adding 8 before the divide:
 ADD    r4, r2, #8 ; add 1/2 resolution
 MOV    r6, r4, LSR #4 ; divide by 16... R6 will have total divisor
 AND    r8, r6, #0xFF  ; copy DLL to r8
 MOV    r9, r6, LSR #8 ; copy DLM to r9
 AND    r9, r9, #0xff ; mask upper bytes
```

36.2.17 Baud Rate Generator

The following table given below is a divisor table for the baud rate, assuming the oscillator is 24.000 MHz.

Table 231: Baud Rate Generator Programming

Baud Rate	Divisor
300	5000
1200	1250
2400	625

Table 231: Baud Rate Generator Programming

Baud Rate	Divisor
4800	312
9600	156
19.2K	78
38.4K	39
57.6K	26
115.2K	13

36.2.18 SIR Pulse Encoder/Decoder

The UART transmit (TX) data is passed through the SIR Pulse Encoder/Decoder module prior to being muxed out to the IR transmitter. This module converts transmitted zeros into a 3/16 Return-To-Zero (RZ) pulse.

Similarly, received (RX) data is bit-synchronized using the 16X baud clock and the original serial data recovered from the IR bit stream. This data is then sent to the UART for reception.

The signal generated in SIR is as follows:

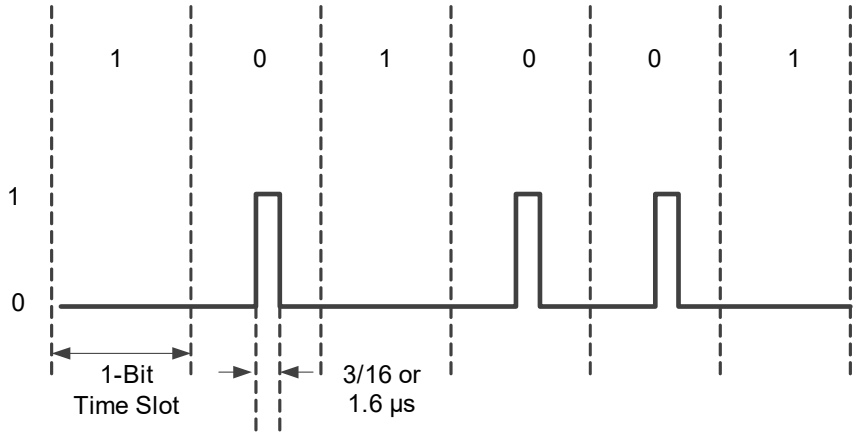
- On logic '1' the LED is off.
- On logic '0' a pulse is created starting the center of the bit time and lasting 3/16 of bit time period or 4/16 of bit time period depending on the current settings.

The figure below displays the output for the input 101001 (in sending order) in SIR encoding.

If SIR encoding/decoding has to be used in loopback mode, the following programming guideline/sequence is required:

1. Program SIR enable
2. Wait for a minimum of (4*baudclk)+ (4clk24m)
3. Program loopback enable

Figure 165: Output in Sending Order in SIR Encoding



36.2.19 UART Interrupts and Interrupt Service Routine (ISR)

There are multiple conditions on which a UART interrupt will be generated. Following are the interrupts available for the Tegra UART controller. Each of the interrupts is routed to an interrupt controller, if the corresponding Interrupt enable bit in the IER register is set.

1. EORD: This indicates an end of received data interrupt. This interrupt is set when UART RX detects there is no data being received for more than 4 character times. This is useful to determine that the transmitter is done with sending all its data. This interrupt is not generated if data is stopped because of handshaking.

2. RXT: RX_TIMEOUT interrupt occurs when data has been sitting in RX_FIFO for more than 4 character times due to some reason. One reason could be because there is not enough data to reach the trigger level. This interrupt needs to be handled by software to have data reception in a timely manner.
3. MSI: Modem status interrupt occurs when one of the CTS/RI/CD/DSR lines change state. The gap between state change for CTS toggle should be at least eight APB clock cycles and four APB cycles for RI/DSR and CD pins to allow internal synchronization and capture it as a delta change for interrupt generation. Each of the lines has a corresponding modem interrupt enable defined in the MIE register.
4. RXS: This interrupt indicates there are errors detected in the received data. The error could be break error/overrun error/parity error or framing error. A Break error occurs when a break condition (all zeros) has been detected on an RX line. An Overrun error indicates a new character reception from an external device when the RX FIFO is already full. Parity error occurs because of parity mismatch and framing error occurs when stop bit of a character received is '0' instead of '1'.
5. THR: This interrupt indicates the Transmit Holding Register is empty. In FIFO mode, this will be set when there is enough space in the FIFO based on trigger level setting so that software can start writing TX data into the THR register.
6. RDR: Receive data ready interrupt is asserted when number of entries in RX_FIFO has reached the trigger level programmed when in FIFO mode. This bit is an indicator to start reading the contents of RX_FIFO.

When uart_intr is asserted, software is expected to read the IIR register to know which interrupt occurred. The interrupts in the UART are prioritized and hence IIR always shows the current highest priority interrupt. If more than one interrupt is pending to be serviced, the one with the highest priority will be shown in IIR. When all interrupts have been serviced, IIR shows no_interrupt. The interrupt priority is defined per the encoding below (the priorities are listed highest to lowest):

IIR[3:0] priority level

- 0001: no interrupt
- 0110: Overrun Error, Parity Error, Framing Error, Break
- 0100: Receiver Data Available
- 1100: rx_timeout_intr
- 1000: eord_timeout_intr
- 0010: Transmitter Holding Register empty
- 0000: modem_status interrupt

The ISR (interrupt service routine) for UART interrupts is not same for all and varies depending on type of interrupt. This section summarizes the ISR expected for clearing each of the interrupt condition.

- RXS interrupt: LSR read clears this interrupt. RXS interrupt conditions include parity/break/overrun or framing error. Parity/break/framing errors also clear on RX FIFO pop operation.
- RDR/RX_TIMEOUT interrupt: These interrupts will be cleared on reading the Read buffer register
- EORD interrupt: IER[5] needs to be disabled to clear this interrupt. The bit can be re-enabled again to receive further interrupts.
- THR interrupt: IIR read or write to buffer data register will clear this interrupt. The interrupt keeps occurring until writes are performed to buffer the write register to fill the FIFO and reach the trigger level.
- Modem status interrupt: This interrupt clears on reading MSR register

Disabling interrupt_enable bit for any of the following interrupts will lead to clearing of uart_intr.

Bit	Reset	Description
3	0x0	IE_MSI: Interrupt Enable for Modem Status Interrupt 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	IE_RXS: Interrupt Enable for Receiver Line Status Interrupt. The interrupt enable for RXS interrupt (IE_RXS) should be disabled whenever LSR is being polled/read to avoid spurious interrupts because of errors detected in RX data. LSR read is an interrupt clearing mechanism for RXS errors. 0 = DISABLE 1 = ENABLE
1	0x0	IE_THR: Interrupt Enable for Transmitter Holding Register Empty interrupt 0 = DISABLE 1 = ENABLE
0	0x0	IE_RHR: Interrupt Enable for Received Data Interrupt 0 = DISABLE 1 = ENABLE

Notes: *In case of an RX_TIMEOUT interrupt, once the interrupt is asserted, disabling only IE_RX_TIMEOUT before reading buffer data register might lead to a spurious case. At least one FIFO pop must have been done before disabling IE_RX_TIMEOUT only. IE_RHR, IE_THR, IE_RXS, or IE_MSI has to be disabled along with IE_RX_TIMEOUT in case IE has to be disabled before an RX FIFO pop operation.*

- *In the UART controller, status registers are also masked with corresponding IE bits unlike conventional controller's implementation.*

36.2.20 FIFO Flushing Guidelines

The UART controller provides TX_CLR and RX_CLR bits to flush the FIFOs. The FIFOs can be flushed independently or simultaneously by following the programming guidelines below:

Non-loopback mode:

- Simultaneous flush (TX_FLUSH and RX_FLUSH in single register write)
  a. Make sure there is no new data being written to TX buffer
  b. Program rts_enable to 0 (MCR[6])
  c. Make sure MCR[1] is 0
  d. Wait for 1 character time
  e. Issue flush requests - Program TX_CLR to 1 and RX_CLR to 1 (UART_IIR_FCR_0[2], UART_IIR_FCR_0[1])
  f. Wait 32 baud cycles
  g. Poll for TMTY to be '1' (UART_LSR_0[6])
  h. Poll for RDR to be '0' (UART_LSR_0[0])
  i. Re-enable rts_enable MCR[6]
  j. Perform new transfers
- Independent flush operations (TX/RX separately)

  TX flush:

  a. Make sure there is no new data being written to TX_buffer
  b. Program TX_CLR to 1 (UART_IIR_FCR_0[2] **-** TX_FLUSH request)
  c. Wait 32 baud cycles
  d. Poll for TMTY to be '1' (UART_LSR_0[6])
  e. Perform new transfers

  RX flush:

a. Program rts_enable to 0 (MCR[6])
b. Make sure MCR[1] is 0
c. Wait for 1 character time
d. Program RX_CLR to 1 (UART_IIR_FCR_0[1] **-** RX_FLUSH request)
e. Poll for RDR to be ‘0’ (UART_LSR_0[0])
f. Re-enable rts_enable MCR[6]
g. Perform new transfers

Loopback mode:

- Simultaneous flush operations do not work (TX_FLUSH and RX_FLUSH in a single register write). The only option is to disable the FIFO mode bit.
- Two independent flushes to be issued as mentioned below to flush both FIFOs:
  a. Program rts_enable to 0 (MCR[6])
  b. Make sure MCR[1] is 0
  c. Program TX_CLR to 1 (UART_IIR_FCR_0[2]**-** TX_FLUSH request)
  d. Wait 32 baud cycles
  e. Poll for TMTY to be ‘1’ (UART_LSR_0[6])
  f. Program RX_CLR to 1 (UART_IIR_FCR_0[1] **-** RX_FLUSH request)
  g. Poll for RDR to be ‘0’ (UART_LSR_0[0])
  h. Re-enable rts_enable MCR[6]
  i. Perform new transfers

36.2.21 Power-up Defaults

The Power-Up defaults are defined below:

- IER = 0
- IIR = 1
- LCR = 0
- MCR = 0
- LSR = 260 HEX
- MSR = Bits 0-3 = 0, Bits 4-7 = inputs line status
- TX = High
- RTS = High
- DTR = High
- RXRDY = Low
- TXRDY = Low
- INT = Low

36.3 UART Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

The Tegra X1 UART is based on the 16550 industry standard Universal Asynchronous Receiver/Transmitter (UART), with enhancements to support autobaud detection and End-of-Received Data timeout detection.

The UART supports a device clock of up to 200 MHz, for a maximum baud rate of 12.5 Megabits per second.

The THR, RBR, and DLL registers all occupy the same address space.

- The Transmitter Holding Register (THR) is Write-Only, and can be accessed if the LSR.DLAB bit is clear.
- The Receiver Buffer Register (RBR) is Read-Only, and can be accessed if the LSR.DLAB bit is clear.
- The Divisor Latch LSByte Register (DLL) is Read/Write and can be accessed if the LSR.DLAB bit is set.

36.3.1 UART_THR_DLAB_0_0

UART Transmit Holding Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
7:0	RO	0x0	THR_A: Transmit holding register, holds the character to be transmitted by the UART. In FIFO mode, a write to this FIFO places the data at the end of the FIFO.
7:0	RO	X	RBR_A: Receive Buffer Register. Rx Data read from here.
7:0	RO	0x0	DLL_A: Divisor Latch LSB (low 8 bits of 16-bit Baud Divisor)

36.3.2 UART_IER_DLAB_0_0

The IER and DLH registers occupy the same address space, selected by the LSR.DLAB bit. The Interrupt Enable Register (IER) is selected if the LSR.DLAB bit is clear. The Divisor Latch MSByte register (DLM) is selected if the LSR.DLAB bit is set. The DLM register holds the upper 8 bits of the 16-bit Baud Divisor (16x).

UART Interrupt Enable por=0x00000000

RX_TIMEOUT occurs when data has been sitting in the Rx FIFO for more than 4 character times without being read because there is not enough data to reach the trigger level. Interrupt needed to handle this case so that all data is received in a timely manner. Note that this normally occurs at the end of an incoming data stream.

EORD (End of Receive Data) Interrupt occurs when the receiver detects that data stops coming in for more than 4 character times. This interrupt is useful for determining that the sending device has completed sending all its data. EORD timeout will not occur if the receiving data stream is stopped because of hardware handshaking.

To clear the EORD timeout interrupt you must DISABLE the EORD interrupt enable (IE_EORD).

Interrupt Enable and Divisor Latch MSByte Registers

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	IE_EORD: Interrupt Enable for End of Received Data 0 = DISABLE 1 = ENABLE
4	0x0	IE_RX_TIMEOUT: Interrupt Enable for Rx FIFO timeout 0 = DISABLE 1 = ENABLE
3	0x0	IE_MSI: Interrupt Enable for Modem Status Interrupt 0 = DISABLE 1 = ENABLE
2	0x0	IE_RXS: Interrupt Enable for Receiver Line Status Interrupt 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	IE_THR: Interrupt Enable for Transmitter Holding Register Empty interrupt 0 = DISABLE 1 = ENABLE
0	0x0	IE_RHR: Interrupt Enable for Received Data Interrupt 0 = DISABLE 1 = ENABLE

36.3.3 UART_IIR_FCR_0

The FCR and IIR registers occupy the same address space. The FIFO Control Register (FCR) is write-only. The Interrupt Identification Register (IIR) is read-only.

UART FIFO Control Register por=0x00000000

The DMA can run in one of two modes:

- If Mode0 is selected (NO_CHANGE), the Rx DMA request goes active whenever the Rx FIFO is not empty. Only single byte transactions can be performed in this mode. If the FIFO is not enabled, Mode0 MUST be used.
- If Mode1 is selected (CHANGE), the Rx DMA request goes active whenever the Rx FIFO trigger level is reached.

For best performance, always enable the FIFO and select DMA Mode 1.

For RX_TRIG, the FIFO_COUNT references the number of bytes in the receive FIFO.

For TX_TRIG, the FIFO_COUNT references the number of empty bytes in the transmit FIFO. For example, FIFO_COUNT_GREATER_16 means that there are at least 16 empty byte slots in the TX_FIFO.

UART Interrupt ID Register por=0x00000001

The Interrupt ID field indicates the current highest priority interrupt. If more than one interrupt is pending the one with the highest priority will be shown.

The table below shows the encoding. Priority flows from top (highest) to bottom (lowest).

Table 232: Interrupt ID Encoding

IIR[3:0]	Priority Level
0001	No interrupt
0110	Overrun Error, Parity Error, Framing Error, Break
0100	Receiver Data Available
1100	rx_timeout_intr
1000	eord_timeout_intr
0010	Transmitter Holding Register empty
0000	modem_status interrupt

UART FIFO Control and Interrupt Identification Registers

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:6	X	EN_FIFO: FIFO Mode Status 0=16450 mode(no FIFO) 1 = 16550 mode (FIFO) 1 = MODE_16550 0 = MODE_16450

Bit	Reset	Description
7:6	0x0	RX_TRIG: 0 = FIFO_COUNT_GREATER_1 1 = FIFO_COUNT_GREATER_4 2 = FIFO_COUNT_GREATER_8 3 = FIFO_COUNT_GREATER_16
5:4	0x0	TX_TRIG: 0 = FIFO_COUNT_GREATER_16 1 = FIFO_COUNT_GREATER_8 2 = FIFO_COUNT_GREATER_4 3 = FIFO_COUNT_GREATER_1
3	X	IS_PRI2: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
3	0x0	DMA: 0:DMA_MODE_0 1:DMA_MODE_1 0 = NO_CHANGE 1 = CHANGE
2	X	IS_PRI1: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
2	0x0	TX_CLR: 1 = Clears the contents of the transmit FIFO and resets its counter logic to 0 (the transmit shift register is not cleared or altered). This bit returns to 0 after clearing the FIFOs. 0 = NO_CLEAR 1 = CLEAR
1	X	IS_PRI0: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
1	0x0	RX_CLR: 1 = Clears the contents of the receive FIFO and resets its counter logic to 0 (the receive shift register is not cleared or altered). This bit returns to 0 after clearing the FIFO. 0 = NO_CLEAR 1 = CLEAR
0	X	IS_STA: Interrupt Pending if ZERO 0 = INTR_PEND 1 = NO_INTR_PEND
0	0x0	FCR_EN_FIFO: 1 = Enable the transmit and receive FIFOs. This bit should be enabled 0 = DISABLE 1 = ENABLE

36.3.4 UART_LCR_0

UART Line Control Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	DLAB: Divisor Latch Access Bit (set to allow programming of the DLH, DLM Divisors) 0 = DISABLE 1 = ENABLE
6	0x0	SET_B: Set BREAK condition -- Transmitter sends all zeroes to indicate BREAK 0 = NO_BREAK 1 = BREAK
5	0x0	SET_P: Set (force) parity to value in LCR [4] 0 = NO_PARITY 1 = PARITY

Bit	Reset	Description
4	0x0	EVEN: Even parity format. There will always be an even number of 1s in the binary representation (PAR = 1) 0 = DISABLE 1 = ENABLE
3	0x0	PAR: 0 = No parity sent 0 = NO_PARITY 1 = PARITY
2	0x0	STOP: 0 = Transmit 1 stop bit 1 = Transmit 2 stop bits (receiver always checks for 1 stop bit) 0 = DISABLE 1 = ENABLE
1:0	0x0	WD_SIZE: 3=Word length of 8 0 = WORD_LENGTH_5 1 = WORD_LENGTH_6 2 = WORD_LENGTH_7 3 = WORD_LENGTH_8

36.3.5 UART_MCR_0

The RTS and CTS pins are used for hardware handshaking with an external serial device. RTS (Request-To-Send) informs the device that the UART is ready to accept data. CTS (Clear-To-Send) comes from the RTS of the external device and indicates that data can be sent.

The RTS pin can be controlled manually with the RTS bit in the MCR or automatically by setting the RTS_EN bit.

If RTS_EN is set, the RTS pin automatically removes the Request-To-Send when the FIFO reaches the trigger level.

If the CTS_EN bit is set, the UART transmitter stops transmitting when the Clear-To-Send input is taken away.

UART Modem Control Register

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx00000000000)

Bit	Reset	Description
10	0x0	OLD_SIR_DECODE: Diagnostics bit to use the required SIR decode path. 0=new SIR decode path, 1=old SIR decode path 0 = DISABLE 1 = ENABLE
9:8	0x0	RI_POLARITY: Polarity selection bit for RI pin toggling to generate modem status interrupt. 0x0 - interrupt will be generated when RI pin toggles from low to high 0x1 - interrupt will be generated when RI pin toggles from high to low 0x2 - interrupt will be generated on RI delta change detection 0x3 – Reserved 0 = LOW_TO_HIGH 1 = HIGH_TO_LOW 2 = BOTH_EDGES 3 = RSVD
7	0x0	DEL_QUAL_CTS_EN: Diagnostics bit to use the old qualified CTS in TX state machine 1=use old qualified_cts 0= use the new qualified_cts in TX state machine 0 = DISABLE 1 = ENABLE
6	0x0	RTS_EN: 1 = Enable RTS Hardware Flow Control 0 = DISABLE 1 = ENABLE
5	0x0	CTS_EN: 1 = Enable CTS Hardware Flow Control 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	LOOPBK: 1 = Enable internal loop back of Serial Out to In 0 = DISABLE 1 = ENABLE
3	0x0	OUT2: nOUT2 (Not Used) 0 = DISABLE 1 = ENABLE
2	0x0	OUT1: nOUT1 (Not Used) 0 = DISABLE 1 = ENABLE
1	0x0	RTS: 0 = Force RTS to high if RTS hardware flow control not enabled 0 = FORCE_RTS_HI 1 = FORCE_RTS_LOW
0	0x0	DTR: 1 = Force DTR to high 0 = FORCE_DTR_HI 1 = FORCE_DTR_LOW

36.3.6 UART_LSR_0

UART Line Status Register

POR = 0x00000260

Offset: 0x14 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	RX_FIFO_EMPTY: 1=RX FIFO is empty 0=RX FIFO has at least one entry 0 = NOT_EMPTY 1 = EMPTY
8	X	TX_FIFO_FULL: Transmitter FIFO full status 0 = NOT_FULL 1 = FULL
7	X	FIFOE: 1 = Receive FIFO Error 0 = NO_ERR 1 = ERR
6	X	TMTY: Transmit Shift Register empty status 0 = NO_EMPTY 1 = EMPTY
5	X	THRE: 1 = Transmit Holding Register is Empty -- OK to write data 0 = FULL 1 = EMPTY
4	X	BRK: 1 = BREAK condition detected on line 0 = NO_BREAK 1 = BREAK
3	X	FERR: 1 = Framing Error 0 = NO_FRAME_ERR 1 = FRAME_ERR
2	X	PERR: 1 = Parity Error 0 = NO_PARITY_ERR 1 = PARITY_ERR
1	X	OVRF: 1 = Receiver Overrun Error 0 = NO_OVERRUN_ERROR 1 = OVERRUN_ERROR
0	X	RDR: 1 = Receiver Data Ready (Data available to read) 0 = NO_DATA_IN_FIFO 1 = DATA_IN_FIFO

36.3.7 UART_MSR_0

UART Modem Status Register

POR = 0x000000f0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	CD: State of Carrier detect pin 0 = DISABLE 1 = ENABLE
6	0x0	RI: State of Ring Indicator pin 0 = DISABLE 1 = ENABLE
5	0x0	DSR: State of Data set ready pin 0 = DISABLE 1 = ENABLE
4	0x0	CTS: State of Clear to send pin 0 = DISABLE 1 = ENABLE
3	0x0	DCD: Change (Delta) in CD state detected 0 = DISABLE 1 = ENABLE
2	0x0	DRI: Change (Delta) in RI state detected 0 = DISABLE 1 = ENABLE
1	0x0	DDSR: Change (Delta) in DSR state detected 0 = DISABLE 1 = ENABLE
0	0x0	DCTS: Change (Delta) in CTS state detected 0 = DISABLE 1 = ENABLE

36.3.8 UART_SPR_0

UART Scratchpad Register

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SPR_A: Scratchpad register (not used internally)

36.3.9 UART_IRDA_CSR_0

Bits [7:6] control the infrared (SIR) encoding. If enabled, each zero bit is transmitted as a short IR pulse. No pulse is sent during idle or '1' data bits. The IR pulse can be either 3/16 or 4/16 of a normal bit time.

The low 4 bits control signal polarity and apply whether or not SIR coding is enabled.

UART IrDA Pulse Coding CSR Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00xx0000)

Bit	Reset	Description
7	0x0	SIR_A: 1 = Enable SIR coder 0 = Disable SIR coder 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	PWT_A: 0=3/16th Baud Pulse, 1=4/16 0 = BAUD_PULSE_3_14 1 = BAUD_PULSE_4_14
3	0x0	INVERT_RTS: Inverts the normally inactive high nRTS pin 0 = DISABLE 1 = ENABLE
2	0x0	INVERT_CTS: Inverts the normally inactive high nCTS pin 0 = DISABLE 1 = ENABLE
1	0x0	INVERT_TXD: Inverts the normally inactive high TXD pin 0 = DISABLE 1 = ENABLE
0	0x0	INVERT_RXD: Inverts the normally inactive high RXD pin 0 = DISABLE 1 = ENABLE

36.3.10 UART_RX_FIFO_CFG_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x000001)

Bit	Reset	Description
7	0x0	EN_RX_FIFO_TRIG: Enable use of RX_FIFO_TRIG count (obsoletes RX_TRIG when enabled) 0 = DISABLE 1 = ENABLE
5:0	0x1	RX_FIFO_TRIG: Set RX_FIFO trigger level (any value 1 thru 32)

36.3.11 UART_VENDOR_STATUS_0_0

UART Controller Status Register

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:24	X	TX_FIFO_COUNTER: The entry in this field reflects the number of current entries in the TX FIFO
21:16	X	RX_FIFO_COUNTER: The entry in this field reflects the number of current entries in the RX FIFO
3	X	TX_OVERRUN: This bit is set to 1 when write data is issued to the TX FIFO when it is already full and gets cleared on register read (sticky bit until read) 0 = NO_OVERRUN 1 = OVERRUN
2	X	RX_UNDERRUN: This bit is set to 1 when a read is issued to an empty FIFO and gets cleared on register read (sticky bit until read) 0 = NO_UNDERRUN 1 = UNDERRUN
1	X	Reserved
0	X	Reserved

36.3.12 UART_ASR_0

The UART has autobaud sensing logic that can measure the width of the start bit of incoming data to determine baud rate. For this to work, the incoming character must have its LSB set. A <cr> works well (0x0d). The logic uses the UART device clock to count how wide the start pulse is, and the BUSY bit is set when the sensing is complete. The value in {RX_RATE_SENSE_H,RX_RATE_SENSE_L} should be divided by 16 with Round-to-Nearest, and the resulting value loaded into the DLM,DLL register pair to set the Baud Clock to 16X the Baud Rate.

UART Auto Sense Baud Register

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	VALID: This bit is set when the controller finishes counting the clocks between two successive clock edges after there is a write to ASR with don't care data. 0 = UN_SET 1 = SET
30	0x0	BUSY: This bit is set when there is a write to ASR and is reset when the controller finishes counting the clock edges between two successive clock edges. 0 = NO_BUSY 1 = BUSY
15:8	0x0	RX_RATE_SENSE_H: Shows bits [15:8] of the count of clock edges between two successive clock edges.
7:0	0x0	RX_RATE_SENSE_L: Shows bits [7:0] of the count of clock edges between two successive clock edges.

CHAPTER 37: SERIAL PERIPHERAL INTERFACE (SPI) CONTROLLER

The Serial Peripheral Interface (SPI) controller is a serial communications link between the processor and on-chip/off-chip serial peripheral devices such as sensors, ADC/DAC devices, and Flash controllers. Tegra® X1 devices support both Master and Slave modes of SPI operation on this interface. There are four SPI controllers in each Tegra X1 device with six pinmuxing options.

In this section:

- *Word* refers to 32-bit data in the TX or RX FIFO
- *Packet* refers to variable length data that is transmitted or received
  - Packet size can be any length from 4 to 32 bits in Master Unpacked mode or 8 to 32 bits in Slave Unpacked mode
  - In Packed Mode, packet size can be 4, 8, 16, or 32 bits

37.1 Functionality

The Tegra X1 SPI Controller works as a Master/Slave on the SPI bus. It has independent transmit and receive FIFOs of 64 x 32 bits each. Software can program the controller to generate transactions of a required packet length on the SPI bus, where a transaction is a sequence of packets in either direction.

The controller supports either APB DMA or program transfer to read and write from the FIFOs as required. At the end of each transaction, an interrupt can be generated, if enabled. Software uses transmit and receive operations in combination with chip select (CS) control to generate commands on the SPI bus.

The interface consists of four signals: Chip Select/Slave Select (CS_N), Clock (SCLK), Master Data Out and Slave Data In (MOSI), and Master Data In and Slave Data Out (MISO).

Features

- Master and Slave functionality
  - Master: All transmission format modes are supported for both transmit and receive operations
  - Slave (transmits): Modes 1 and 3 are supported
  - Slave (receives): All modes are supported
- Independent Rx and Tx FIFOs
- FIFO depth of 64 x 32 bits
- Programmable bit lengths, resulting in packet sizes of 4 to 32 bits
- PIO or DMA Mode depending on packet size

  Options include Packed/Unpacked, Full-Duplex Mode, Both Enable Bit (x2 Mode), Bidirectional, Least Significant Bit, Least Significant Byte First (Endian-ness), software or hardware chip-select polarity selection
- Programmable clock phase and polarity
- Programmable delay between consecutive transfers
- Packed mode support for bit lengths of 3 (4-bit packet size), 7 (8-bit packet size), and 15 (16-bit packet size)
- Chip select (CS) can be controlled by software or can be generated automatically by hardware on packet boundaries
- Maximum 4-chip support with programmable CS polarity for each chip select
- DMA support
- Simultaneous receive and transmit operations supported

- The maximum data rate is 65 Mbps in master mode and 45 Mbps in slave mode.

37.1.1 Transmission Format

Using the Mode field in the SPI Command1 Register, software sets one of the four modes in which the SPI controller works. The two bits of the Mode field specify the idle (inactive) state of SCLK before the chip select is asserted and the data transmitting/ receiving edges of SCLK. Mode [1] determines the SCLK polarity. If the polarity is 0, then the SCLK signal is 0 when idle and transitions to 1 when data transfer starts. If the clock polarity is 1, then the SCLK signal is 1 when idle and transitions to 0 when data transfer starts. Mode [0] is the SCLK phase control bit. If the clock phase is 0, then the receiver latches the data on the first transition of SCLK from the idle state. If the clock phase is 1, then the receiver latches the data on the second transition of SCLK.

Modes supported by the SPI controller are:

- Master: All transmission format modes are supported both for transmitting and receiving
- Slave: Modes 1 and 3 are supported for transmitting
- Slave: All modes are supported for receiving

The following table defines the four modes for the SPI protocol.

Table 233: SPI Modes with Clock Polarity and Phase

SPI Mode	Clock Polarity	Clock Phase	SCLK Inactive State	Data Latch IN	Data Latch OUT
0	0	0	Low	Latched IN on the positive edge of clock	Latched OUT on the negative edge of clock
1	0	1	Low	Latched IN on the negative edge of clock	Latched OUT on the positive edge of clock
2	1	0	High	Latched IN on the negative edge of clock	Latched OUT on the positive edge of clock
3	1	1	High	Latched IN on the positive edge of clock	Latched OUT on the negative edge of clock

37.1.2 Modes of Operation

37.1.2.1 Master and Slave Modes

The SPI controller can be configured to operate as a Master or Slave. When the M/S bit in the SPI Command1 Register is set, Master mode is selected; when the M/S bit in the SPI Command1 Register is cleared, Slave mode is selected. Only the SPI Master can initiate a transaction.

In Master mode, data from the Tx FIFO is transmitted on the MOSI pin. The data from the slave is received on the MISO pin and sent to the Rx FIFO. The Master can simultaneously transmit and receive on MOSI and MISO in Full-Duplex Mode.

In Slave mode, once software enables the SPI controller by setting the PIO bit, it simply waits for the Master to initiate a transaction. Before the transaction begins, the slave logic continuously polls the CS input. When the Master asserts CS and SCLK is transmitted to the Slave, the Slave data is transmitted from the Tx FIFO on MISO and data from MOSI is received in the Rx FIFO. The Slave can also simultaneously transmit and receive on MOSI and MISO in Full-Duplex Mode.

Table 234: Master Mode Port Configuration

Name	Direction	Description
MOSI	Bidirectional	Output in Full-Duplex/Tx Mode Input in Both_Enable Rx Mode
MISO	Bidirectional	Input in Full-Duplex/Rx Mode
SCLK	Output	Output in Both_Enable TX Mode
CS_#x	Output	Slave select signal to x, where x is a number between 0 to 3

Table 235: Slave Mode Port Configuration

Name	Direction	Description
MISO	Bidirectional	Output in Full-Duplex/Tx Mode Input in Both_Enable Rx Mode
MOSI	Bidirectional	Input in Full-Duplex/Rx Mode Output in Both_Enable TX Mode
SCLK	Input	Synchronization clock received from the Master
CS_#	Input	Select signal

When the SPI controller is configured to interface with multiple slaves, the controller has one CS signal for each slave, up to a maximum of four slaves. During a transfer, the master asserts CS specified in the CS_SEL bits in SPI Command1 Register. There can be no more than one slave transmitting data during any particular transfer.

37.1.2.2 DMA and PIO Modes

PIO and DMA are the two main operational modes for the SPI controller. PIO mode is used when the number of packets to the transmitter or receiver is less than or equal to 64. DMA mode is used when the number of packets to the transmitter or receiver is greater than 64. In PIO or DMA mode, the SPI controller sends an interrupt to the processor at the end of each transfer. In this mode software configures the number of packets and number of bits in each packet to be transmitted/received. The limitation of PIO Mode is that it can be used only if the maximum number of packets that can be transferred is less than or equal to 64 packets (the FIFO depth is 64). If software wants to transmit/receive more than 64 packets in PIO Mode, it has to reconfigure the SPI controller every 64 packets.

To send/receive more than 64 packets, the SPI Controller can be configured in DMA mode (enabled by writing 1 to the DMA_EN bit in the SPI DMA Control Register). The SPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the SPI Block Size Register. BLOCK_SIZE is a 16-bit field, so 2^{16} packets (amounting to 256 KB) can be transmitted/received in a single software configuration.

37.1.2.3 Continuous Mode

Continuous Mode means:

- In Master Mode, the SPI Controller can transfer data continuously with an external device, unless interrupted by software.
- In Slave Mode, the SPI Controller can transfer data continuously with an external device, unless interrupted by software or an external Master de-asserts chip select (CS). In the CS de-assert case, the SPI controller can generate an interrupt based on intr mask bit.

Notes:

- *This mode can be enabled only in DMA mode, NOT in PIO mode. The SPI Controller will not consider the "DMA_BLOCK_SIZE" register in this mode.*
- *To perform two consecutive back-to-back continuous mode transfers, the SPI controller must be reset after the first continuous mode transfer before the subsequent continuous mode transfer can occur.*
- *In SPI slave continuous mode, the last data packet is not pushed into the RX FIFO if CS has been de-asserted at the 0xffff(+1) boundary by the external master. The last packet will be pushed into the slave buffer when CS comes back the next time; otherwise it is lost.*

SPI Controller Configured as Master

When the SPI Controller is configured as a Master, it can transfer any amount of data unless software disables the SPI Controller by clearing (write 1 to clear) the DMA_EN bit. The SPI Controller generates CS and Sclk accordingly throughout the data transfer. Along with the SPI Controller, the APB DMA also needs to be configured in Continuous Mode. Refer to the APB section in this TRM for more information.

- Errors and Interrupt generation:

  In Master Continuous Mode, the SPI controller will not generate TX_FIFO Underrun or RX_FIFO Overrun errors. These errors generally occur in slave mode when the APB DMA does not write data into the TX_FIFO or does not read data from the RX_FIFO fast enough due to system bandwidth/arbitration issues. In such cases, the SPI Controller keeps waiting for the APB DMA to write data into the TX_FIFO or read data from the RX_FIFO. During such waiting periods (Idle periods), the SPI Controller stops generating Sclk but keeps CS asserted. Once data is available in the TX_FIFO or enough space is available in RX_FIFO, the SPI Controller resumes the data transfer and generates Sclk, continuing the CS in the asserted state.

- Disabling the SPI Controller in Master Continuous Mode:

  In Master Continuous Mode, the SPI Controller can be disabled only by clearing (write 1 to clear) the DMA_EN bit by software. The SPI Controller will not be disabled in any other way by hardware.

  - Generally software is recommended to clear (write 1 to clear) the DMA_EN bit when the Controller is in the Idle state.
  - If software clears the DMA_EN bit during the Active state (when a transaction is in process, i.e., CS is asserted and Sclk is present), the SPI Controller will write the last received partial/full word (whatever is present in the shift register) up to the point before disabling the DMA_EN bit into the RX_FIFO and increments the blk_count_received register.

SPI Controller Configured as Slave

When the SPI Controller is configured as a Slave, it can transfer any amount of data unless:

- Software disables the SPI Controller by clearing the DMA_EN bit (write 1 to clear) (or)
- External SPI Master Device de-asserts Chip Select.
- External SPI Master Stops sending Sclk for SLV_IDLE_COUNT No.of Clocks.

Errors and Interrupt Generation

In Slave Continuous Mode, the SPI controller will generate TX_FIFO Underrun or RX_FIFO Overrun errors. These errors generally occur when APB DMA does not write data into the TX_FIFO or does not read data from RX_FIFO fast enough due to system bandwidth/arbitration issues. In such cases, the SPI Controller generates an error condition, sets the TX_FIFO_URN or RX_FIFO_OVR bits, and generates an interrupt. If an Overflow happens, data in the RX_FIFO will not be corrupted.

Disabling the SPI Controller in Slave Continuous Mode:

In Slave Continuous Mode, the SPI Controller can be disabled by software clearing the DMA_EN bit.

Generally, software is recommended to clear the DMA_EN bit when the SPI Controller is in the Idle state (the Idle state is defined as when there is no input Sclk to the Tegra SPI slave from the External Master device and/or CS is inactive).

If software clears the DMA_EN bit during the Active state (the Active state is defined as when there is an input Sclk to the Tegra SPI slave from the External Master device and CS is active), the SPI Controller will write the last received partial/full word (whatever is present in shift register) into the RX_FIFO and increments blk_count_received register.

Disabling the SPI Controller by an External Master Device

When the SPI Controller is configured in Slave Continuous Mode, the External Master can start a data transfer by generating CS and Sclk to the Slave SPI. Once the External Master finishes the data transfer (tx, rx, or both), it will stop generating Sclk and then de-assert CS. The SPI Slave treats this as an end of transaction and generates an Interrupt along with clearing the DMA_EN bit. The SPI Controller will write the last received partial/full word (whatever is present in the shift register before CS is de-asserted) into the RX_FIFO and increments the blk_count_received register. This will be true even if CS is deasserted in an unaligned word boundary (i.e., CS deassertion can happen only at a packet boundary).

In Slave Error cases (RX_FIFO_OVR), the blk_count_received register is incremented first and then the RX_FIFO is written. So when RX_FIFO_OVR happens, the blk_count_received also shows 1 word extra compared to what is actually present in the RX_FIFO.

The SPI Controller will not generate a CS_deassert interrupt at the very end when the actual programmed blk_size is done. That time CS is de-asserted by the master normally. Software will get a done interrupt in this case.

CS_INACTIVE Bit in SPI FIFO Control/Status Register

In the CS de-assert case, the SPI controller can generate interrupts provided the interrupt Mask bit is set low by the software and sets the CS_INACTIVE bit of SPI FIFO Control/Status Register indicating CS is de-asserted by an External Master. Software has to clear this bit while addressing the interrupt. If interrupt Mask bit is set high, the CS_INACTIVE bit of the SPI FIFO Control/Status Register will still be set high by the SPI Controller, but the interrupt is not generated.

Notes:

- *If the CS deassert interrupt mask is set low (i.e., the interrupt is enabled), the SPI Slave Controller generates an interrupt if CS is de-asserted and all the data in the RX shift register, until the point CS is de-asserted, is written into the RX_FIFO.*
- *If the CS deassert interrupt mask is set high (i.e., the interrupt is disabled), the SPI Slave Controller does not generate an interrupt if CS is de-asserted and treats it as a PAUSE condition and it DOES NOT write all the data in the RX shift register into the RX_FIFO until the CS is asserted back.*
- *In both the above cases, CS deassertion Interrupt generation is independent of the Clock provided by the External Master.*

FRAME_END in SPI FIFO Control/Status Register

In Slave Continuous Mode, if Sclk is not received for the number of clocks specified in the SLV_IDLE_COUNT field and CS is asserted, then the continuous mode is terminated and this status bit is set. Continuous Mode is terminated if the condition is met. In this case, an interrupt is generated.

Currently, software has two use cases:

- In Slave Continuous mode, software mainly will use the pause feature (mask the CS de-assert interrupt), because the use case will be such that there will be continuous audio steaming and it is not desired for the slave to be stopped based on CS de-assert.
- In Slave non-continuous mode, where the use case is based on variable length data coming in from the external master, software will use the non-pause feature (enable the CS de-assert interrupt) so that software can read the received chunk of data based on the CS de-assert interrupt.

In the Non-Pause case, continuous mode termination can happen with CS deasserted or DMA disabled.
In the Pause case, continuous mode termination can happen only with DMA disable. Until the DMA is disabled, it is the same continuous mode transaction and continues forever until the DMA is disabled.

37.1.2.4 Packed and Unpacked Modes

SPI communication can be carried out in Packed or Unpacked Mode. These are both present in DMA and PIO Modes.

- Packed Mode: In Packed Mode, data can be transmitted/received with each packet size equal to 4, 8, 16, or 32 bits. The PACKED field and BIT_LEN field in the DMA Command1 Register determine if the mode Packed or Unpacked and the length of each packet. If the BIT_LEN field is set to 03h, each packet is 4 bits long and each word in the Tx/Rx FIFO contains 8 such packets. Similarly, if BIT_LEN is 07h, the packet length is 8. If BIT_LEN is 0Fh, the packet length is 16. If BIT_LEN is 1Fh, the packet length is 32. In Packed Mode, if BIT_LEN is set to 03h and if the number of packets is set to a non-multiple of 8 (for example, 9), the last word in FIFO contains only the ninth packet with 4 valid bits; 28 extra invalid bits will be ignored by the controller for transmits and will contain invalid data for receives.
- Unpacked Mode: If BIT_LEN is set to 03h, then each word in the Tx/Rx FIFO has 4 bits with the remaining bits in each word being ignored. BIT_LEN can be any value from 3 to 31 in Unpacked Mode.

37.1.2.5 Bidirectional Mode

Bidirectional mode is selected when the BIDIR bit is set in the SPI Command1 Register. In this mode, the SPI controller uses only one serial data pin for the interface with external device(s). The M/S bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the Master Mode, and the MISO pin becomes serial data I/O (SISO) pin for the Slave Mode. The SPI controller does not use the MISO pin in Master Mode and the MOSI pin in Slave Mode.

Table 236: Bidirectional Mode Port Configuration for Master (M/S Bit Set)

Name	Direction	Description
MOSI > MOMI	Bidirectional	Data Output and Input to Slave(s)
MISO > --	--	--
SCLK	Output	Synchronization clock to all Slaves
CS	Output	Slave select signal to x, where x is a number from 0 to 3, inclusive

Table 237: Bidirectional Mode Port Configuration for Slave (M/S Bit Cleared)

Name	Direction	Description
MOSI > --	--	--
MISO > SISO	Bidirectional	Data Output and Input from Slave(s)
SCLK	Input	Synchronization clock
CS	Input	Select signal

37.1.2.6 Both Enable Bit Mode (SPI x2 Mode)

The Both Enable Bit Mode is selected when the BOTH_EN_BIT bit is set in the SPI Command1 Register. In Both Enable Bit Mode, a packet from the Tx FIFO is transmitted or received on both MOSI and MISO data lines. Even bits are transmitted or received on the MOSI data line and odd bits are transmitted or received on the MISO data line. This is the same in both Master and Slave Modes.

37.1.2.7 En_LE_Bit and En_LE_Byte Modes

Note: *En_LE_Bit is Enable Little Endian Bit, and En_LE_Byte is Enable Little Endian Byte.*

These two bits allow a data packet to be transmitted or received in Little Endian or Big Endian format. Once a data packet is read from the Tx FIFO, it undergoes a two-step pre-processing based on En_LE_Byte and En_LE_Bit.

Step 1. In this step, En_LE_Byte configures whether a packet has to be arranged internally according to the Little Endian byte format or Big Endian byte format. For example, if a 2-byte packet is written into the FIFO by software as Byte 1,Byte 0, setting En_LE_Byte = 0 makes the SPI controller arrange the packet internally as Byte 0,Byte 1(Big Endian format). If En_LE_Byte = 1, the SPI controller internally arranges the packet as Byte 1,Byte 0 (Little Endian format) before the packet is transmitted.

Step 2. In this step, the output of Step 1 is taken, and data is put on the MOSI/MISO line according to En_LE_Bit. If En_LE_Bit is set to 0, the most significant bit of the output of Step 1 is put on the MOSI/MISO line first. If En_LE_Bit is set to 1, the least significant bit of Step 1 output is put on the MOSI/MISO line.

These steps are illustrated in the following table.

Step 1	FIFO Packet (MSB - LSB)	Byte	Output of Step 1
	Byte 1, Byte 0	0	Byte 0, Byte 1
		1	Byte 1, Byte 0

Step 1	FIFO Packet (MSB - LSB)	Byte	Output of Step 1
Step 2	Output of Step 1	Bit	Data Transmitted on MOSI (Right-most Bit Sent First)
	Byte 0, Byte 1	0	Byte 1 (0), Byte 1 (1), …, Byte 0 (6), Byte 0 (7)
	Byte 0, Byte 1	1	Byte 0 (7), Byte 0 (6), …, Byte 1 (1), Byte 1 (0)
	Byte 1, Byte 0	0	Byte 0 (0), Byte 0 (1), …, Byte 1 (6), Byte 1 (7)
	Byte 1, Byte 0	1	Byte 1 (7), Byte 1 (6), …, Byte 0 (1), Byte 0 (0)

Similarly for Receive operations, the first packet received will be internally rearranged first based on En_LE_Bit, then the output of the previous step will be rearranged based on En_LE_Byte before writing the final packet into the Rx FIFO.

The following examples explain the effect of these two bits:

- Unpacked Mode
  - Tx: The following example shows how a 20-bit length packet from the FIFO is transmitted. The right-most bit in Column D in the table below is transmitted first. B.7-B.0 indicates that bit 0 is transmitted first followed by bit 1 until bit 7, which is transmitted last.

A	B	C	D
FIFO WORD: 20 (MSB) – 0 (LSB)	En_LE_Bit	En_LE_Byte	Data Transmitted on MOSI/MISO Line
	0	0	16, 17,18, 19, 20, B.8-B.15, B.0-B.7
	0	1	B.0-B.7, B.8-B.15, 16, 17, 18, 19, 20
	1	0	B.7-B.0, B.15-B.8, 20, 19, 18, 17, 16
	1	1	B.20-B.0

  - Rx: The following example shows how a 20-bit length packet from the FIFO is received. The right-most bit in Column D in the table below is received last on the MOSI/MISO lines. B.7-B.0 indicates that bit 7 is received first followed by bit 6 until bit 0, which is received last. B.20 – B.0 are always written into the FIFO after the En_LE_Byte and En_LE_Bit transformations.

A	B	C	D
FIFO WORD: 20 (MSB) – 0 (LSB)	En_LE_Bit	En_LE_Byte	Data Received on MOSI/MISO Line
	0	0	B.7-B.0, B.15-B.8, 20, 19, 18, 17, 16
	0	1	20, 19, 18,17, 16, B.15-B.8, B.7-B.0
	1	0	16, 17, 18, 19, 20, B.8-B.15, B.0-B.7
	1	1	B.0-B.20

- Packed Mode
  - Tx: The following example shows how 3 16-bit packets are transmitted. Column B indicates the valid packets to be transmitted. Because FIFO Word2 is not transmitted entirely, valid packets are only from B.15-B.0. In Column D, P1(0 -15) indicates Packet1 of Column B is transmitted, B.15 is transmitted first, and B.0 is transmitted last.

A	B: Valid Packets	B	C	D
FIFO WORD1: 31 (MSB) – 0 (LSB)	P1(15-0), P0(15-0)	En_LE_Bit	En_LE_Byte	Data Transmitted on MOSI/MISO Line
FIFO WORD2: 31 (MSB) – 0 (LSB)	P2(15-0)	0	0	P2(8-15), P2(0-7), P1(8-15), P1(0-7), P0(8-15), P0(0-7)
		0	1	P2(0-7), P2(8-15), P1(0-7), P1(8-15), P0(0-7), P0(8-15)
		1	0	P2(7-0), P2(15-8), P1(7-0), P1(15-8), P0(7-0), P0(15-8)

A	B: Valid Packets	B	C	D
		1	1	P2(15-8), P2(7-0), P1(15-8), P1(7-0), P0(15-8), P0(7-0)

- Rx: The following example shows how 3 16-bit packets are received. Column B indicates the valid received packets in the Rx FIFO. In column D, P1(0 -15) indicates Packet1 of Column B is received, B.0 is received first, and B.15 is received last.

A	B: Valid Packets	B	C	D
FIFO WORD1: 31 (MSB) – 0 (LSB)	P1(15-0), P0(15-0)	En_LE_Bit	En_LE_Byte	Data Received on MOSI/MISO Line
FIFO WORD2: 31 (MSB) – 0 (LSB)	P2(15-0)	0	0	P0(7-0), P0(15-8), P1(7-0), P1(15-8), P2(7-0), P2(15-8)
		0	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)
		1	0	P0(8-15), P0(0-7), P1(8-15), P1(0-7), P2(8-15), P2(0-7)
		1	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)

37.2 SPI Programming Guidelines

There are two basic modes of operation: DMA and PIO modes. It is required that software sets up all parameters in the SPI Command1 and Command2 registers, SPI CS Timing 1 and 2 registers, SPI FIFO Control/Status register, SPI DMA Control register, and SPI Block Size register before enabling the PIO bit in any of these modes.

37.2.1 DMA Mode

This mode is enabled by writing 1 to the DMA bit in the SPI DMA Control register. In this mode, the SPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the SPI Block Size register.

If the PACKED bit is set and BIT_LEN is set to 7, then all FIFO words contain 4 packets to transfer (transmit or receive). Packets will be transferred as per the En_LE_Bit and En_LE_Byte bit configurations (see the En_LE_Bit and En_LE_Byte Modes section), with packet 0 in byte 0 of the FIFO and packet 3 in byte 3 of the FIFO.

In Unpacked Mode, if BIT_LEN is set to N, each packet will consist of (N + 1) bits. These bits will be transmitted/received in the Tx_FIFO/Rx_FIFO as per the En_LE_Bit and En_LE_Byte bit configurations (see the En_LE_Bit and En_LE_Byte Modes section). Any remaining bits in the FIFO will be ignored by the hardware. The maximum packet length is 32, which can be selected by setting BIT_LEN to 31. In this case, all data bits in the FIFO contain valid packet data.

A DMA request will be generated to APB_DMA in this mode depending on the setting of Tx_TRIG and Rx_TRIG. If transmits are enabled, setting Tx_TRIG to 00 will generate a Tx DMA request whenever the Tx FIFO has one word of space available (if not full). Setting Tx_TRIG to 01 will generate a Tx DMA request whenever the Tx FIFO has 4 words of space available. If receives are enabled, setting Rx_TRIG to 00 will generate an Rx DMA request whenever the Rx FIFO has one word of data available (is not empty). Setting Rx_TRIG to 01 will generate an Rx DMA request whenever the Rx FIFO has 4 words of data available.

37.2.2 PIO Mode

This mode is enabled by writing 1 to the PIO bit in the SPI Command1 register. In this mode, the SPI controller transmits/ receives as many packets as configured by software in the SPI Block Size Register.

Note: *Software must wait for the RDY bit, whether in polling mode or interrupt mode, for confirmation of transfer completion.*

PIO Mode has the same features as DMA Mode. The difference between PIO Mode and DMA Mode is that in PIO Mode, the maximum number of packets that can be transmitted or received is less than or equal to 64.

37.2.3 Interrupt Generation

The SPI controller generates an interrupt to the processor at the end of a transfer in PIO Mode or when an error is detected (both in PIO and DMA Modes) if the IE.TX or IE.RX bit in the SPI DMA Control register is enabled for transmit and receive modes of operation, respectively.

If IE.TX is enabled during transmits, an interrupt is generated whenever RDY or one of the FIFO status bits in the SPI FIFO Control/Status Register is set by the hardware. During transmits, when the SPI controller is configured as a Slave, if software/APB DMA cannot fill the transmit FIFO fast enough, hardware will set the Tx_FIFO_UNR bit and an underrun condition is generated. If software tries to write to a full Tx FIFO, hardware sets the Tx_FIFO_OVF bit as an indication that software attempted to overflow the Tx FIFO. Hardware makes sure that the overflowing data is never written to the Tx FIFO.

If IE.RX is enabled during receives, an interrupt is generated whenever RDY or one of the status bits in the SPI FIFO Control/Status Register is set by the hardware. During receives, when the SPI controller is configured as a Slave, if software/APB DMA cannot read the receive FIFO fast enough, hardware will set the Rx_FIFO_OVF bit and an overflow condition is generated. However, if software tries to read from an empty Rx FIFO, hardware sets the RXF_UNR bit as an indication that software attempted to underrun the Rx FIFO.

The interrupt can be cleared by writing a 1 to the source of the interrupt. If the interrupt is generated by assertion of RDY, then writing a 1 to the RDY bit clears the interrupt. If the interrupt is generated by assertion of Tx_FIFO_OVF, then writing a 1 to the TXF_OVF bit clears the interrupt. If the interrupt is generated by assertion of Rx_FIFO_UNR, then writing a 1 to RXF_UNR bit clears the interrupt.

Guideline for DMA use cases: In DMA Mode, if interrupts are enabled, the SPI software driver has to deal with 3 interrupts: one from the SPI controller and two from the Tx and Rx APB DMA channels. This might make it complicated for the driver and also stresses the system. In such cases, software can use the Rx DMA interrupt as the final one at which point all hardware activities can be assumed to be complete. But, in variable transfer lengths (continuous mode), software has to rely on the controller's interrupts.

37.2.4 Clock Initialization and Control

The SPI controller runs at 50 MHz at its interface to external SPI devices. The internal SPI controller clock (spi_clk) should run at the same frequency as the outgoing Interface Clock (Sclk_Out) in Master Mode. In Slave Mode, the internal SPI interface clock frequency has to be 50% greater than the external clock (Sclk_in).

The SPI clock is selected from different clock sources using a generic clock switch module. The clock switch module supports glitch-free switching of clock from one source to another. There are 4 clock sources:

- osc_clk
- pllP_out
- pllC_out
- pllM_out0

The selected clock is divided by an 8-bit value written in the divider register. The clock is then trimmed, and gated branches are generated for the SPI instances.

> ***Note:*** *The ratio between the slave core clock and the slave interface clock must be maintained between 1.5x and 4x. For example if the slave interface clock is running at 10 MHz, the slave core clock must not be less than 10x1.5 MHz and not greater than 10x4 MHz.*

The CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0, CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0, and CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0 registers in the CAR (Clock and Reset) block contain the enable bits for the SPI controller. In addition, the clock source and divider registers (CLK_SOURCE_SBC1, CLK_SOURCE_SBC2,

CLK_SOURCE_SBC3, CLK_SOURCE_SBC4, CLK_SOURCE_SBC5, and CLK_SOURCE_SBC6) contain the 2-bit field SBCx_CLK_SRC to determine the PLL clock source, while the SBCx_CLK_DIVISOR field determines the divide ratio.

See Chapter 5: Clock and Reset Controller for more information on clock configuration.

Controller Reset

The SPI controller can be reset by writing 1 to SPI1 (bit 11) in the RST_DEVICES.H (0x6000:6008) register. Software needs to write a 0 to this bit to bring the SPI controller out of reset.

See the Clock and Reset Controller section for more information on SPI controller reset.

Slave Mode GPIO Synchronization

Because SPI does not support flow control, the GPIO is required to have the proper communication to avoid any clock loss from the Master. GPIO can be used in the following way for proper synchronization:

- SPI slave client software will write to the Config registers of the SPI controller and make the SPI slave ready.
- After that the slave client software will toggle GPIO to inform the Master that the Slave is ready.
- Then the SPI master sends SCLK to the slave device.

If there is no GPIO to tell the Master that the Slave is ready, there is a chance to lose clock/data.

Guidelines

This section provides guidelines for programming the SPI controller:

- Program all the required register fields (no particular order is required except for the PIO/DMA bit, which has to be programmed last):
  - Clock Mode
  - Packed/Unpacked Mode
  - Tx_EN/Rx_EN
  - BOTH_EN_BIT
  - En_LE_Bit/En_LE_Byte
  - BIT_LEN and BLOCK_SIZE values
  - CS_SW_HW (software based or hardware based)
    - If CS_SW_HW is set, the value on CS_SW_VAL will be driven out.
    - When CS HW is used program the Setup & Hold Values in CS Timing Register.
  - Program the DMA Trig values appropriately, if DMA Mode is used
  - Enable interrupts if PIO Mode is used
- Lastly, program PIO or DMA to indicate the start of transfer.

Once software enables PIO/DMA, no required bits can be changed until the end of transfer except for PIO/DMA bit. Clearing the PIO/DMA bit will end the transaction. In case the SPI controller is configured to Receive Mode and software clears the PIO/DMA bit, then the partial data which the controller received until then will be written into the FIFO. This is true both for Master and Slave.

Do not enable conflicting features. For example, avoid enabling BOTH_EN_BIT along with bidirectional (BIDIR) mode.

In Slave Mode after PIO or DMA is set, hardware first waits for the CS to go low by one of the 4 Masters. Then on reception of Sclk, transfer begins. If CS or Sclk is removed by the Master in the middle of a transfer, software must terminate the transaction by clearing the PIO/DMA bit. In case of Rx, hardware then writes the last received packet (incomplete packet) to the Rx FIFO. If software does not clear the PIO/DMA bit, the SPI controller will just keep waiting for the Sclk from the Master. If Sclk resumes later, the transaction continues from where it paused earlier.

If any error conditions occur, hardware will set the corresponding status bits in the SPI FIFO Control/Status register and stop the transfer. If IE is enabled, an interrupt is generated. Software has to clear the source of the interrupt by writing a 1 to it, after the completion of ISR.

Special Guidelines for Slave Mode

Software should not program the controller register when the interface clock is toggling. Software can disable the external clock input before programming the registers and then re-enable it once all the registers bits are programmed.

Error Conditions

When the SPI controller is configured as a Master, the following error scenarios can happen:

- Tx_FIFO overflow

  Tx_FIFO overflow happens when the CPU/APB_DMA writes into the Tx FIFO when it is full. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_OVF bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO overflow

  Rx_FIFO overflow happens when the CPU/APB_DMA writes into the Rx FIFO when it is full in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Rx_FIFO_OVF bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Tx_FIFO underrun

  Tx_FIFO underrun happens when the CPU/APB_DMA reads from the Tx FIFO when it is empty. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_UNR bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO underrun

  Rx_FIFO underrun happens when the CPU/APB_DMA reads from the Rx FIFO in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Rx_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_UNR bit, along with the Interrupt bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

Note: *In Master Mode, all the errors above can happen only when the CPU/APB_DMA reads/writes an empty/full FIFO. The SPI controller will never write/read a full/empty FIFO. Instead the controller will pause (stops sending clocks to the slave with chip select being in an active state) whenever the FIFOs are full/empty.*

When the SPI controller is configured as a Slave, the following error scenarios can happen:

- Tx_FIFO overflow

  Tx_FIFO overflow happens when the CPU/APB_DMA or the SPI controller writes into the Tx FIFO when it is full. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Tx_FIFO_OVF bit in the status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and Tx_FIFO_OVF bit. To start a new transaction, software has to disable the Master from sending clocks and then flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO overflow

Rx_FIFO overflow happens when the CPU/APB_DMA or the SPI controller writes into Rx FIFO when it is full in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Rx_FIFO_OVF bit in the status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and Rx_FIFO_OVF bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Tx_FIFO underrun

  Tx_FIFO underrun happens when the CPU/APB_DMA or the SPI controller reads from Tx FIFO when it is empty. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the TX_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and TX_FIFO_UNR bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO underrun

  Rx_FIFO underrun happens when the CPU/APB_DMA or the SPI controller reads from the Rx FIFO in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the RX_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and RX_FIFO_UNR bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- CS_inactive

  When the SPI controller is configured as a Slave and cs_active between packets is set, if an external Master deasserts the chip select in the middle of a transaction, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and sets the CS_INACTIVE bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the CS_INACTIVE bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

  It is software's responsibility to read the data that is left out of the RX FIFO for CS inactive. For example, if the interrupt trigger is set for 4 words, and external master has taken the CS off in between the transfer, and, at that point of time, the RX FIFO has only 2 words, it is software's responsibility to read out those 2 words or flush the FIFO.

- Frame_End

  When the SPI controller is configured as a Slave and DMA-continuous and cs_active between packets is set, if an external Master stops sending clocks for more than slave_idle_clock_count programmed in the register, the SPI controller will end the transaction by setting the DMA bit to 0, and sets the FRAME_END bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the FRAME_END bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register. If the Master resumes the clk within "slave_idle_clock_count", the SPI controller will also pause and resume the transaction whenever ext_clk is available.

  There are three scenarios for continuous mode pause:

  Assuming SLV_IDLE_CNT = 255:

  - Master stops sending the clock but the CS is asserted for > 255 cycles, frame end is asserted.
  - Master stops sending the clock when CS is asserted for 100 cycles, then CS is de-asserted for some time. After the CS is asserted again, master does not send the clocks for > 155 cycles then frame end is asserted.
  - Master stops sending the clock when CS is asserted for 100 cycles and then CS is de-asserted for some time. After the CS is asserted again, master resumes sending the clock before 155 cycles, then frame end is not asserted.

Programming Trimmers

The 3 programmable trimmers in the SPI controller are used only in Master Mode:

- SPI-Tx trimmer

This trimmer is used in Master Tx mode to adjust/center the outgoing data with respect to the outgoing clock.

- SPI-Rx trimmer

  This trimmer is used in Master Rx mode to delay the loopback clock to the Rx shift registers. This trimmer is used to adjust the Master SCLK with respect to the SPI slave device's Tx data.

- SPI-Spare Control Register Byte1

  This 3-bit trimmer controls the internal timing of the Master mode Rx datapath. This trimmer along with the SPI Rx trimmer may be set independently to adjust the loopback clock delay. This trimmer adjusts the programmable delay on the u_spi_slave_rx* flops in inbound paths. These are half cycle paths, and delays have been added to increase Setup margin on these paths.

- The SPI controller has a programmable delay chain with a 3-tap delay chain. The 3 taps use 100D, 100D, and 60D and are programmed as follows:
  - 000: Bypass through each mux: will see 3 muxes in path
  - 001: 60D
  - 010: 100D
  - 100: 100D
  - 011: 60D + 100D
  - 110: 100D + 100D
  - 101: 100D+60D
  - 111: 100D+100D+60D
- Spare cells used as control for these delay taps:
  - (u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_2_,
  - u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_1_,
  - u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_0_)

37.3 SPI Controller Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

There are four sets of the SPI controller registers, one for each of the four SPI controllers. The register descriptions below give the offset of each register within its SPI controller's address range. See Chapter 2: Address Map for the starting address of each SPI controller.

37.3.1 SPI_COMMAND_0

SPI Command1 Register

This register is used to set the bit length and to select the transfer mode.

Chip Select can also be selected to be in hardware mode as software mode with both active high and active low polarities to support devices with varying CS polarities.

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x43d00000 (0b01000011110100000000xxxxx000000)

Bit	Reset	Description
31	0x0	PIO: Program 1 after all the other bits in the SPI_COMMAND2 and SPI_COMMAND1 registers are programmed to start the transfer. Hardware clears this bit automatically after the transfer is done. Clearing of this bit by software will stop the shifter and latch the partial data into the buffer (in Receive Mode). 0 = STOP (default) 1 = GO

Bit	Reset	Description
30	0x1	M/S: Master/Slave mode select. 0 = Slave Mode (external clock) 1 = Master Mode (internal clock) (default)
29:28	0x0	MODE: The SPI interface clock mode has to be programmed according to the device with which it is communicating. 0 = Mode 0 (default) 1 = Mode 1 2 = Mode 2 3 = Mode 3 Master: All modes are supported both for Tx and Rx. Slave: Modes 1and 3 are supported for Tx. Slave: All modes are supported for Rx.
27:26	0x0	CS_SEL: In Master Mode, these bits are programmed to select a slave in the multi-slave environment. 0 = Selects CS0 (default) 1 = Selects CS1 2 = Selects CS2 3 = Selects CS3
25	0x1	CS_POL_INACTIVE#3: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS3, needs to be programmed. 1 = CS3 Inactive value of External device is high, 0 = CS3 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
24	0x1	CS_POL_INACTIVE#2: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS2, needs to be programmed, 1 = CS2 Inactive value of External device is high, 0 = CS2 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
23	0x1	CS_POL_INACTIVE#1: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS1, needs to be programmed. 1 = CS1 Inactive value of External device is high, 0 = CS1 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
22	0x1	CS_POL_INACTIVE#0: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS0, needs to be programmed. 1 = CS0 Inactive value of External device is high, 0 = CS0 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
21	0x0	CS_SW_HW: Software control of the SPI_CS signal in Master Mode. In Slave Mode, this bit need not be programmed. 1 = CS controlled by software; 0 = CS controlled by hardware 0 = SPI _CS#(CS_SEL) is driven to the active state during packet transfers by the hardware (default) 1 = SPI _CS#(CS_SEL) is driven with the value in the CS_SW_VAL bit
20	0x1	CS_SW_VAL: CS signal value in Master Mode. In Slave Mode, this bit need not be programmed. If CS_SW_HW is 1, then the value in CS_SW_VAL is driven out on CS. If CS_SW_HW is 0, then this bit has no effect. 0 = CS is LOW 1 = CS is HIGH (default)
19:18	0x0	IDLE.SDA: Inactive data signal format. Controls the output enable of the SDA line when the controller is inactive and not doing data transfers. 0 = Drive low. (Master Mode) (default) 1 = Drive high. (Master Mode) 2 = External Pull Down. (Slave Mode) 3 = External Pull high. (Slave Mode)
17	0x0	BIDIR: Bidirectional Transfer Control Bit: This bit enables the bidirectional mode of the transfer. 0 = Normal Mode (default) 1 = Bidirectional Mode
16	0x0	En_LE_Bit: 1 = Enable Little Endian Bit. 0 = Disable Little Endian Bit 0 = LAST 1 = FIRST
15	0x0	En_LE_Byte: 1 = Enable Little Endian Byte. 0 = Disable Little Endian Byte 0 = LAST 1 = FIRST

Bit	Reset	Description
14	0x0	BOTH_EN_BIT (x2 Mode): Both MISO and MOSI can also be used to transmit or receive at the same time when this bit is set. Even bits from the Tx FIFO packet are transmitted/ received on MOSI, and Odd bits from the Tx FIFO packet are transmitted/received on MISO. Refer to the Both Enable Bit Mode (SPI x2 Mode) section for more information. 0 = Normal Mode (default) 1 = Transmit/Receive on both MISO and MOSI at the same time when the Tx_EN and Rx_EN bits are set
12	0x0	Rx_EN: Receive enable. Setting this bit to 1 enables the controller to receive the data. 0 = DISABLE (default) 1 = ENABLE
11	0x0	Tx_EN: Transmit enable. Setting this bit to 1 enables the controller to transmit the data. 0 = DISABLE (default) 1 = ENABLE
10:6	N/A	Reserved for future use. Always write zeros.
5	0x0	PACKED: Packed mode enable bit.. 0 = Unpacked Mode (default) 1 = Packed Mode. This is only valid if the BIT_LEN field is set to either 3 (4-bit transfer), 7 (8-bit transfer), 15 (16-bit transfer), or 31 (32-bit transfer).
4:0	0x0	BIT_LEN: This field is used during PIO and DMA transfers. It represents the number of bits transmitted/received in either Packed or Unpacked Mode. The minimum bit length in Master Mode is 4 and the minimum length in Slave Mode is 1 byte (BIT_LEN = 7). The default is 0. 3 = 4-bit transfer N = N+1-bit transfer 31 = 32-bit transfer

37.3.2 SPI_COMMAND2_0

SPI Command2 Register

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
31:12	N/A	Reserved for future use. Always write zeros.
11:6	0x0	Tx_Clk_TAP_DELAY: Delays the clock going out to the external device with these tap values. Useful only in Master Mode.
5:0	0x0	Rx_Clk_TAP_DELAY: Delays the clock coming in from the external device with these tap values. Useful only in Master Mode.

37.3.3 SPI_TIMING_REG1_0

SPI CS Timing1 Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:28	0x0	CS_SETUP_TIME#3: Specifies the setup time of the chip select to the data being transmitted or received on CS#3 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
27:24	0x0	CS_HOLD_ TIME#3: Specifies the hold time of the chip select to the data being transmitted or received on CS#3 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
23:20	0x0	CS_SETUP_ TIME#2: Specifies the setup time of the chip select to the data being transmitted or received on CS#2 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
19:16	0x0	CS_HOLD_ TIME#2: Specifies the hold time of the chip select to the data being transmitted or received on CS#2 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

Bit	Reset	Description
15:12	0x0	CS_SETUP_ TIME#1: Specifies the setup time of the chip select to the data being transmitted or received on CS#1 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
11:8	0x0	CS_HOLD_ TIME#1: Specifies the hold time of the chip select to the data being transmitted or received on CS#1 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
7:4	0x0	CS_SETUP_ TIME#0: Specifies the setup time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
3:0	0x0	CS_HOLD_ TIME#0: Specifies the hold time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

37.3.4 SPI_TIMING_REG2_0

SPI CS Timing2 Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x20202020 (0bxx100000xx100000xx100000xx100000)

Bit	Reset	Description
31:30	N/A	Reserved for future use. Always write zeros.
29	0x1	CS_ACTIVE_BETWEEN_PACKETS#3: Specifies if CS stays active between two packets on CS#3 1 = CS active between two packets (default) 0 = CS inactive between two packets
28:24	0x0	CYCLES_BETWEEN_PACKETS#3: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#3. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
23:22	N/A	Reserved for future use. Always write zeros.
21	0x1	CS_ACTIVE_BETWEEN_PACKETS#2: Specifies if CS stays active between two packets on CS#2 1 = CS active between two packets (default) 0 = CS inactive between two packets
20:16	0x0	CYCLES_BETWEEN_PACKETS#2: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#2. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
15:14	N/A	Reserved for future use. Always write zeros.
13	0x1	CS_ACTIVE_BETWEEN_PACKETS#1: Specifies if CS stays active between two packets on CS#1 1 = CS active between two packets (default) 0 = CS inactive between two packets
12:8	0x0	CYCLES_BETWEEN_PACKETS#1: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#1. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
7:6	N/A	Reserved for future use. Always write zeros.
5	0x1	CS_ACTIVE_BETWEEN_PACKETS#0: Specifies if CS stays active between two packets on CS#0 1 = CS active between two packets (default) 0 = CS inactive between two packets

Bit	Reset	Description
4:0	0x0	CYCLES_BETWEEN_PACKETS#0: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#0. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

37.3.5 SPI_TRANSFER_STATUS_0

SPI TRANSFER Status Register

Read this register for the status of the transfer.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00ff0000 (0bx0xxxxx11111111xxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	N/A		Reserved for future use. Always write zeros.
30	RW	0x0	RDY: Ready bit. This bit is set to 1 at the end of every transfer, and an interrupt is also generated if the corresponding interrupt enable is set in PIO/DMA Mode. Software writes a 1 to clear it. The interrupt is also cleared when this bit is cleared. 0 = NOT_READY (default) 1 = READY
29:24	N/A		Reserved for future use. Always write zeros.
23:16	RW	0xff	SLV_IDLE_COUNT: Slave Continuous Mode. If Sclk is not received for this number of cycles, then Slave Continuous mode is terminated, and the status bit FRAME_END is set. The default is 255 (8'hff).
15:0	RO	X	BLOCK_COUNT: Counts the number of packets in a transaction (Tx or Rx) in DMA/PIO Mode. In DMA Continuous Mode, this field gives the number of packets only if the number of packet is less than 2^{16}.

37.3.6 SPI_FIFO_STATUS_0

SPI Control/Status FIFO Status Register

SPI STATUS register: Read this register to know the status of the transfer. Error bit is set whenever errors such as Underflow/ Overflow are encountered.

Offset: 0x14 | Read/Write: R/W | Reset: 0x00400005 (0b00xxxxxxxxxxxxxx00xxxxx00000xxxx)

Bit	R/W	Reset	Description
31	RW	0x0	CS_INACTIVE: When the SPI is in Slave Mode, if CS is deasserted, this bit is set and the received number of bits is written in the Rx FIFO. An interrupt is generated if IE.Tx or IE.Rx is set. The default is 0.
30	RW	0x0	FRAME_END: When the SPI is in Slave Continuous mode, if Sclk is not received for the number of clocks specified in the SLV_IDLE_COUNT field, the continuous mode is terminated and this status bit is set. The default is 0.
29:23	RO	X	RX_FIFO_FULL_COUNT: Indicates the number of slots in the receive FIFO remaining before the FIFO is empty. This field is used by software for debugging purposes.
22:16	RO	X	TX_FIFO_EMPTY_COUNT: Indicates the number of slots in the transmit FIFO remaining before the FIFO is full. This field is used by software for debugging purposes.
15	RW	0x0	RX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Rx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH
14	RW	0x0	TX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Tx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH
13:9	N/A		Reserved for future use. Always write zeros.
8	RW	0x0	ERR: Will be set to 1 by hardware when errors such as underflow/overflow occur. Software writes a 1 to clear the flag. 0 = OK (default) 1 = ERROR

Bit	R/W	Reset	Description
7	RW	0x0	TX_FIFO_OVF: TX FIFO Overflow. This bit is set to 1 whenever the SPI controller or software tries to write to a full Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TX in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
6	RW	0x0	TX_FIFO_UNR: TX FIFO Underrun. This bit is set to 1 whenever the SPI controller or software tries to read from an empty Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
5	RW	0x0	RX_FIFO_OVF: RX FIFO Overflow. This bit is set to 1 whenever the SPI controller or software tries to write into a full Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
4	RW	0x0	RX_FIFO_UNR: RX FIFO Underrun. This bit is set to1 whenever the SPI controller or software tries to read from an empty Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
3	RO	X	TX_FIFO_FULL: TX FIFO Full Status. Hardware sets this bit to 1 if the Tx FIFO is full; otherwise this bit is 0. The Tx FIFO is empty at power on reset (POR). 0 = NOT_FULL (default) 1 = FULL
2	RO	X	TX_FIFO_EMPTY: TX FIFO Empty Status. Hardware sets this bit to 1 if the Tx FIFO is empty; otherwise this bit is 0. The Tx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)
1	RO	X	RX_FIFO_FULL: RX FIFO Full Status. Hardware sets this bit to 1 if the Rx FIFO is full; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_FULL (default) 1 = FULL
0	RO	X	RX_FIFO_EMPTY: RX FIFO Empty Status. Hardware sets this bit to 1 if the Rx FIFO is empty; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)

37.3.7 SPI_TX_DATA_0

SPI Transmit Data Register

Offset: 0x18 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SPI_Tx_DATA. Transmit Data. This register holds the last data that was transmitted by the SPI controller.

37.3.8 SPI_RX_DATA_0

SPI Receive Data Register

Offset: 0x1c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SPI_Rx_DATA. Receive Data. This register holds the last data that was received by the SPI controller.

37.3.9 SPI_DMA_CTL_0

SPI DMA Control Register

SPI DMA Control Register: Used in the DMA transfers. We can set trigger levels in this register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxx00xx00xxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	DMA: Enable DMA Mode transfer. Software writes a 1 to this bit to start a transfer in the DMA mode. All fields in the SPI_COMMAND1 and SPI_DMA_CTL register must be set before writing a 1 to this bit. This bit is cleared by the SPI controller after all packets have been transferred as indicated by the DMA_BLOCK_SIZE field. Clearing this bit by software will stop the shifter and latch the partial data into buffer. 0 = DMA Mode is disabled (default) 1 = DMA Mode is enabled
30	0x0	CONT: Enable Continuous Mode transfer. 0 = Continuous Mode is disabled (default) 1 = Continuous Mode is enabled
27:21	N/A	Reserved for future use. Always write zeros.
20:19	0x0	RX_TRIG: Receive FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is at least 1 packet in the RX FIFO. (default) 01: 4 words. DMA trigger is asserted when there are at least 4 packets in the RX FIFO. 10: 8 words. DMA trigger is asserted when there are at least 8 packets in the RX FIFO. 11: 16 words. DMA trigger is asserted when there are at least 16 packets in the RX FIFO. (Presently, the APB DMA does not support this trigger level.)
18:17	N/A	Reserved for future use. Always write zeros.
16:15	0x0	TX_TRIG: Transmit FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is space for at least 1 packet in the TX FIFO. (default) 01: 4 words. DMA trigger is asserted when there is space for at least 4 packets in the TX FIFO. 10: 8 words. DMA trigger is asserted when there is space for at least 8 packets in the TX FIFO. 11: 16 words. DMA trigger is asserted when there is space for at least 16 packets in the TX FIFO. (Presently, the APB DMA does not support this trigger level.)

37.3.10 SPI_DMA_BLK_SIZE_0

SPI DMA Block Size Register

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	BLOCK_SIZE: Size of data block to be transferred in PIO/DMA Mode. The default is 0. In DMA Mode, the maximum number of 32-bit packets that can be transferred is 2^{16} = 65536. In PIO Mode, the maximum number of 32-bit packets that can be transferred is 64 (FIFO depth). The Block Size has to be programmed 1 packet lower than intended; for example, a value of 3 indicates 4 packets are to be transferred.

37.3.11 SPI_TX_FIFO_0

SPI TX FIFO Buffer Register

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SPI Tx_FIFO. Tx FIFO.

37.3.12 SPI_RX_FIFO_0

SPI RX FIFO Buffer Register

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SPI Rx_FIFO. Rx FIFO.

37.3.13 SPI_INTR_MASK_0

SPI Interrupt Mask Register

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0b0000000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CS_INTR_MASK: Interrupt enable mask bit for CS deassertion in Slave Mode. 0 = Enable interrupt generation if CS is deasserted in the middle of a transaction in Slave mode. 1 = Disable interrupt generation if CS is deasserted in the middle of a transaction in Slave mode. 0 = DISABLE 1 = ENABLE
30	0x0	FRAME_END_INTR_MASK: Interrupt enable mask bit for Frame End in Slave Mode. 0 = Enable interrupt generation if Frame End is set in the middle of a transaction in Slave mode. 1 = Disable interrupt generation if Frame End is set in the middle of a transaction in Slave mode. 0 = DISABLE 1 = ENABLE
29	0x0	RDY_INTR_MASK: Interrupt enable mask bit for RDY. 0 = Enable interrupt generation when RDY bit is asserted. 1 = Disable interrupt generation when RDY bit is asserted. 0 = DISABLE 1 = ENABLE
28	0x0	TX_FIFO_OVF_INTR_MASK: Interrupt enable mask bit for TX_FIFO_OVF. 0 = Enable interrupt generation when TX_FIFO_OVF is asserted. 1 = Disable interrupt generation when TX_FIFO_OVF is asserted. 0 = DISABLE 1 = ENABLE
27	0x0	TX_FIFO_UNF_INTR_MASK: Interrupt enable mask bit for TX_FIFO_UNF. 0 = Enable interrupt generation when TX_FIFO_UNF is asserted. 1 = Disable interrupt generation when TX_FIFO_UNF is asserted. 0 = DISABLE 1 = ENABLE
26	0x0	RX_FIFO_OVF_INTR_MASK: Interrupt enable mask bit for RX_FIFO_OVF. 0 = Enable interrupt generation when RX_FIFO_OVF is asserted. 1 = Disable interrupt generation when RX_FIFO_OVF is asserted. 0 = DISABLE 1 = ENABLE
25	0x0	RX_FIFO_UNF_INTR_MASK: Interrupt enable mask bit for RX_FIFO_UNF. 0 = Enable interrupt generation when RX_FIFO_UNF is asserted. 1 = Disable interrupt generation when RX_FIFO_UNF is asserted. 0 = DISABLE 1 = ENABLE

37.3.14 SPI_SPARE_CTLR

SPI Spare Control Register

Offset: 0x190 | Read/Write: R/W | Reset: 0x0fff0000 (0b00001111111111111000000000000000)

Bit	Reset	Description
31:11	00001111111111100000b	Reserved for future use.
10:8	000b	SPI_SPARE_CONTROL_REGISTER_BYTE2. Program bits 8, 9, and 10 along with Rx_Clk_TAP_DELAY in the COMMAND2 register to adjust the clock delay on internal registers. Useful in Master Mode only.

Bit	Reset	Description
7:0	0x00	Reserved for future use.

CHAPTER 38: QUAD SERIAL PERIPHERAL INTERFACE (QSPI) CONTROLLER

The Quad Serial Peripheral Interface (Quad SPI) controller is a multi-bit serial communications link between the processor and on-chip/off-chip Serial Flash controllers. The QSPI controller works on the Serial Protocol – Serial Peripheral Interface (SPI). It has configurable features such as:

- Master
- Clock Polarity and Phase (Mode 0)
- DDR or SDR mode
- Dual or Quad mode

In this section:

- *Word* refers to 32-bit data in the TX or RX FIFO
- *Packet* refers to variable length data that is transmitted or received
  - In Packed Mode, packet size can be 8, 16, or 32 bits

38.1 Functionality

The Tegra® X1 QSPI Controller works only as a Master on the QSPI bus. It has independent transmit and receive FIFOs of 64 x 32 bits each. Software can program the controller to generate transactions of a required packet length on the QSPI bus, where a transaction is a sequence of packets in either direction.

The QSPI controller uses APB DMA to read and write from the FIFOs as required. At the end of each transaction, an interrupt can be generated, if enabled. Software uses transmit and receive operations in combination with chip select (CS) control to generate commands on the QSPI bus.

The QSPI controller is a sync client of APB. QSPI registers are in APB clk (pclk) domain. Internal data FIFOs are async FIFOs.

The interface consists of four signals:

- Chip Select (QSPI_CS_N)
- Serial Clock (QSPI_SCK)
- Master Data Out and Slave Data In (MOSI) / QSPI_IO0 for Dual or Quad commands
- Master Data In and Slave Data Out (MISO) / QSPI_IO1 for Dual or Quad commands

Features

- Master and Slave functionality
  - Master: Only Mode 0 is supported for both transmit and receive operations
  - Slave: Not supported
- Independent Rx and Tx FIFOs
- FIFO depth of 64 x 32 bits
- Byte-aligned bit-length in packed and unpacked mode. (BIT_LEN =7/15/31)
- PIO or DMA Mode depending on packet size

  Options include Packed/Unpacked (bit lengths of 7, 15, 31 are supported), Dual mode (x2 Mode), Quad mode (x4 mode), DDR mode (Double Data Rate), Bidirectional (in SDR mode only), Least Significant Bit, Least Significant Byte First (Endian-ness), software or hardware chip-select polarity selection

- Programmable clock phase and polarity
- Packed mode support for bit lengths of 7 (8-bit packet size), 15 (16-bit packet size), and 31 (32-bit packet size)
- In packed mode, it is expected that a minimum of 1 word will be transferred. Software can achieve less than 1 word transfer using unpacked mode.
- Only 1-chip support with programmable CS polarity for chip select
- DMA support
- Simultaneous receive and transmit operations supported
- The maximum frequency of the device clock is 166 MHz (SDR) and 83 MHz (DDR)

38.1.1 Transmission Format

Using the Mode field in the QSPI Command1 Register, software sets one of the two modes in which the QSPI controller works. The two bits of the Mode field specify the idle (inactive) state of SCLK before the chip select is asserted and the data transmitting/receiving edges of SCLK. Mode [1] determines the SCLK polarity. If the polarity is 0, then the SCLK signal is 0 when idle and transitions to 1 when a data transfer starts. If the clock polarity is 1, then the SCLK signal is 1 when idle and transitions to 0 when data transfer starts. Mode [0] is the SCLK phase control bit. If the clock phase is 0, then the receiver latches the data on the first transition of SCLK from the idle state. If the clock phase is 1, then the receiver latches the data on the second transition of SCLK.

Modes supported by the QSPI controller are:

- Master: Mode 0 is supported both for transmitting and receiving
- Slave: Not supported

The following table defines the mode for the QSPI protocol.

Table 238: QSPI Mode with Clock Polarity and Phase

QSPI Mode	Clock Polarity	Clock Phase	SCLK Inactive State	Data Latch IN	Data Latch OUT
0	0	0	Low	Latched IN on the positive edge of clock	Latched OUT on the negative edge of clock

38.1.2 Modes of Operation

38.1.2.1 Master Mode

The QSPI controller operates as a Master only. Only the QSPI master can initiate a transaction.

In Master x1 mode, data from the Tx FIFO is transmitted on the MOSI/IO0 pin. The data from the slave is received on the MISO/IO1 pin and sent to the Rx FIFO.

Similarly in x2 TX mode, the master transmits data on the IO0 and IO1 lines from the TX FIFO and in x2 RX mode, the same IO0 and IO1 lines are used to receive data and push into RX FIFO.

For x4 mode, the IO0 to IO3 lines are used to transmit and receive the data.

Table 239: QSPI Master Mode Port Configuration

Name	Direction	Description
MOSI / IO0	Bidirectional	Output in Tx Mode Input in Rx Mode
MISO / IO1	Bidirectional	Output in Tx Mode Input in Rx Mode
IO2	Bidirectional	Output in Tx Mode Input in Rx Mode

Table 239: QSPI Master Mode Port Configuration

Name	Direction	Description
IO3	Bidirectional	Output in Tx Mode Input in Rx Mode
SCLK	Output	Synchronization clock to slave
CS0#	Output	Slave select signal

38.1.2.2 DMA and PIO Modes

PIO and DMA are the two main operational modes for the QSPI controller. PIO mode is used when the number of packets to the transmitter or receiver is less than or equal to 64. DMA mode is used when the number of packets to the transmitter or receiver is greater than 64. In PIO or DMA mode, the QSPI controller sends an interrupt to the processor at the end of each transfer. In this mode software configures the number of packets and number of bits in each packet to be transmitted/received. The limitation of PIO Mode is that it can be used only if the maximum number of words that can be transferred is less than or equal to 64 words (the FIFO depth is 64). If software wants to transmit/receive more than 64 words in PIO Mode, it has to reconfigure the QSPI controller every 64 words.

To send/receive more than 64 words, the QSPI Controller can be configured in DMA mode (enabled by writing 1 to the DMA_EN bit in the QSPI DMA Control Register). The QSPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the QSPI Block Size Register. BLOCK_SIZE is a 16-bit field, so 2^{16} packets (amounting to a maximum 256 KB if pkt_size = 32 bits) can be transmitted/received in a single software configuration.

CS Usage Guideline

If a hardware-based CS is used, the CS goes to the inactive state after the transfer of packets as programmed in the BLOCK_SIZE register field is completed. If CS has to stay active in multiple consecutive transactions (for example, the first transaction – Tx: 64 Packets; the second transaction – Rx: 128 Packets), then the software based CS has to be used (in hardware-based CS, CS goes to the inactive state after the completion of the first transaction).

> ***Note:*** *Hardware-based CS operates only on Flash devices that support it; otherwise data mismatches may occur. Please refer to the definition of CS in your Flash device's specification.*

38.1.2.3 Packed and Unpacked Modes

SPI communication can be carried out in Packed or Unpacked Mode. These are both present in DMA and PIO Modes.

- Packed Mode: In Packed Mode, data can be transmitted/received with each packet size equal to 8, 16, or 32 bits. The PACKED field and BIT_LEN field in the DMA Command1 Register determine if the mode is Packed or Unpacked and the length of each packet. If the BIT_LEN field is set to 07h, each packet is 8 bits long and each word in the Tx/Rx FIFO contains 8 such packets. If BIT_LEN is 0Fh, the packet length is 16. If BIT_LEN is 1Fh, the packet length is 32. In Packed Mode, if BIT_LEN is set to 07h and if the number of packets is set to a non-multiple of 4 (for example, 5), the last word in FIFO contains only the fifth packet with 8 valid bits; 24 extra invalid bits will be ignored by the controller for transmits and will contain invalid data for receives. In packed mode, a minimum transfer will be 1 word. For less than 1 word transfers, software can use Unpacked mode.

> ***Notes:***
>
> - *When the QSPI controller is programmed with BIT_LEN =7/15 in Packed mode and BLK_SIZE is 0xffff, software should use the RDY interrupt to indicate when the transfer is done instead of using the count in the BLK_CNT field in QSPI_TRANSFER_STATUS register.*
> - *The QSPI controller does not push data into the Rx FIFO in Packed mode with block size = 0xFFFF for x2 and x4 and bit length = 32.*

- Unpacked Mode: The main difference between Packed and Unpacked Mode is that in Packed Mode if BIT_LENGTH is set to 07h then each word in the Tx/Rx FIFO has 8 packets with 8 bits each, whereas in Unpacked Mode each word in Tx/Rx FIFO has (BIT_LEN + 1) bits with the remaining bits word being ignored.

38.1.2.4 Bidirectional Mode

Bidirectional mode is selected when the BIDIR bit is set in the QSPI Command1 Register. Bidirectional mode is possible only in x1 mode. In this mode, the IO0 line is used for both transmit and receive operations. When the BIDIR bit is set with the TX_EN bit, the IO0 line is in output mode to transmit data to external flash. And when BIDIR bit is set with RX_EN, the IO0 line is in input mode to receive data from external flash. Bidirectional mode is supported only in SDR mode.

Table 240: Bidirectional Mode Port Configuration (M/S Bit Set

Name	Direction	Description
MOSI (IO0)	Bidirectional	Data Output and Input to Slave(s)
MISO (IO1)	--	--
SCLK	Output	Synchronization clock to all Slaves
CS	Output	Slave select signal

38.1.2.5 En_LE_Bit and En_LE_Byte Modes

Note: *En_LE_Bit is Enable Little Endian Bit, and En_LE_Byte is Enable Little Endian Byte.*

These two bits allow a data packet to be transmitted or received in Little Endian or Big Endian format. Once a data packet is read from the Tx FIFO, it undergoes a two-step pre-processing based on En_LE_Byte and En_LE_Bit.

Step 1. In this step, En_LE_Byte configures whether a packet has to be arranged internally according to the Little Endian byte format or Big Endian byte format. For example, if a 2-byte packet is written into the FIFO by software as Byte 1,Byte 0, setting En_LE_Byte = 0 makes the QSPI controller arrange the packet internally as Byte 0,Byte 1(Big Endian format). If En_LE_Byte = 1, the QSPI controller internally arranges the packet as Byte 1,Byte 0 (Little Endian format) before the packet is transmitted.

Step 2. In this step, the output of Step 1 is taken, and data is put on the MOSI/MISO line according to En_LE_Bit. If En_LE_Bit is set to 0, the most significant bit of the output of Step 1 is put on the MOSI/MISO line first. If En_LE_Bit is set to 1, the least significant bit of Step 1 output is put on the MOSI/MISO line.

These steps are illustrated in the following table.

Step 1	FIFO Packet (MSB - LSB)	En_LE_Byte	Output of Step 1
	Byte 1, Byte 0	0	Byte 0, Byte 1
		1	Byte 1, Byte 0
Step 2	**Output of Step 1**	**En_LE_Bit**	**Data Transmitted on MOSI (Right-most Bit Sent First)**
	Byte 0, Byte 1	0	Byte 1 (0), Byte 1 (1), …, Byte 0 (6), Byte 0 (7)
	Byte 0, Byte 1	1	Byte 0 (7), Byte 0 (6), …, Byte 1 (1), Byte 1 (0)
	Byte 1, Byte 0	0	Byte 0 (0), Byte 0 (1), …, Byte 1 (6), Byte 1 (7)
	Byte 1, Byte 0	1	Byte 1 (7), Byte 1 (6), …, Byte 0 (1), Byte 0 (0)

Similarly for Receive operations, the first packet received will be internally rearranged first based on En_LE_Bit, then the output of the previous step will be rearranged based on En_LE_Byte before writing the final packet into the Rx FIFO.

The following examples explain the effect of these two bits:

- Unpacked Mode
  - Tx: The following example shows how a 20-bit length packet from the FIFO is transmitted. The right-most bit in Column D in the table below is transmitted first. B.7-B.0 indicates that bit 0 is transmitted first followed by bit 1 until bit 7, which is transmitted last.

A	B	C	D
FIFO WORD: 20 (MSB) – 0 (LSB)	En_LE_Bit	En_LE_Byte	Data Transmitted on MOSI/MISO Line
	0	0	16, 17,18, 19, 20, B.8-B.15, B.0-B.7
	0	1	B.0-B.7, B.8-B.15, 16, 17, 18, 19, 20
	1	0	B.7-B.0, B.15-B.8, 20, 19, 18, 17, 16
	1	1	B.20-B.0

- Rx: The following example shows how a 20-bit length packet from the FIFO is received. The right-most bit in Column D in the table below is received last on the MOSI/MISO lines. B.7-B.0 indicates that bit 7 is received first followed by bit 6 until bit 0, which is received last. B.20 – B.0 are always written into the FIFO after the En_LE_Byte and En_LE_Bit transformations.

A	B	C	D
FIFO WORD: 20 (MSB) – 0 (LSB)	En_LE_Bit	En_LE_Byte	Data Received on MOSI/MISO Line
	0	0	B.7-B.0, B.15-B.8, 20, 19, 18, 17, 16
	0	1	20, 19, 18,17, 16, B.15-B.8, B.7-B.0
	1	0	16, 17, 18, 19, 20, B.8-B.15, B.0-B.7
	1	1	B.0-B.20

- Packed Mode
  - Tx: The following example shows how three 16-bit packets are transmitted. Column B indicates the valid packets to be transmitted. Because FIFO Word2 is not transmitted entirely, valid packets are only from B.15-B.0. In Column D, P1(0 -15) indicates Packet1 of Column B is transmitted, B.15 is transmitted first, and B.0 is transmitted last.

A	B: Valid Packets	B	C	D
FIFO WORD1: 31 (MSB) – 0 (LSB)	P1(15-0), P0(15-0)	En_LE_Bit	En_LE_Byte	Data Transmitted on MOSI/MISO Line
FIFO WORD2: 31 (MSB) – 0 (LSB)	P2(15-0)	0	0	P2(8-15), P2(0-7), P1(8-15), P1(0-7), P0(8-15), P0(0-7)
		0	1	P2(0-7), P2(8-15), P1(0-7), P1(8-15), P0(0-7), P0(8-15)
		1	0	P2(7-0), P2(15-8), P1(7-0), P1(15-8), P0(7-0), P0(15-8)
		1	1	P2(15-8), P2(7-0), P1(15-8), P1(7-0), P0(15-8), P0(7-0)

  - Rx: The following example shows how three 16-bit packets are received. Column B indicates the valid received packets in the Rx FIFO. In column D, P1(0 -15) indicates Packet1 of Column B is received, B.0 is received first, and B.15 is received last.

A	B: Valid Packets	B	C	D
FIFO WORD1: 31 (MSB) – 0 (LSB)	P1(15-0), P0(15-0)	En_LE_Bit	En_LE_Byte	Data Received on MOSI/MISO Line
FIFO WORD2: 31 (MSB) – 0 (LSB)	P2(15-0)	0	0	P0(7-0), P0(15-8), P1(7-0), P1(15-8), P2(7-0), P2(15-8)
		0	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)
		1	0	P0(8-15), P0(0-7), P1(8-15), P1(0-7), P2(8-15), P2(0-7)
		1	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)

38.2 QSPI Programming Model

There are two basic modes of operation: DMA and PIO modes. It is required that software sets up all parameters in the QSPI Command1 and Command2 registers, QSPI CS Timing 1 and 2 registers, QSPI FIFO Control/Status register, QSPI DMA Control register, and QSPI Block Size register before enabling the PIO bit in any of these modes.

38.2.1 DMA Mode

This mode is enabled by writing 1 to the DMA bit in the QSPI DMA Control register. In this mode, the QSPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the QSPI Block Size register.

Choose the INTERFACE_WIDTH to select among x1, x2, and x4 modes. Select SDR_DDR_SEL based on the external QSPI slave device.

If the PACKED bit is set and BIT_LEN is set to 7, then all FIFO words contain 8 packets to transfer (transmit or receive). Packets will be transferred as per the En_LE_Bit and En_LE_Byte bit configurations (see the En_LE_Bit and En_LE_Byte Modes section), with packet 0 in byte 0 of the FIFO and packet 3 in byte 3 of the FIFO.

In Unpacked Mode, if BIT_LEN is set to N, each packet will consist of (N + 1) bits. These bits will be transmitted/received in the Tx_FIFO/Rx_FIFO as per the En_LE_Bit and En_LE_Byte bit configurations (see the En_LE_Bit and En_LE_Byte Modes section). Any remaining bits in the FIFO will be ignored by the hardware. The maximum packet length is 32, which can be selected by setting BIT_LEN to 31. In this case, all data bits in the FIFO contain valid packet data.

A DMA request will be generated to APB_DMA in this mode depending on the setting of Tx_TRIG and Rx_TRIG. If transmits are enabled, setting Tx_TRIG to 00 will generate a Tx DMA request whenever the Tx FIFO has one word of space available (if not full). Setting Tx_TRIG to 01 will generate a Tx DMA request whenever the Tx FIFO has 4 words of space available. If receives are enabled, setting Rx_TRIG to 00 will generate an Rx DMA request whenever the Rx FIFO has one word of data available (is not empty). Setting Rx_TRIG to 01 will generate an Rx DMA request whenever the Rx FIFO has 4 words of data available.

Note: *In DMA mode, the APB DMA and QSPI controllers must be programmed for the same amount of data for correct operation. Otherwise the QSPI controller could fetch too much data.*

38.2.2 PIO Mode

This mode is enabled by writing 1 to the PIO bit in the QSPI Command1 register.

PIO Mode has the same features as DMA Mode. The difference between PIO Mode and DMA Mode is that in PIO Mode, the maximum number of packets that can be transmitted or received is less than or equal to 64.

The QSPI controller is expected to have all the data in FIFOs before software enables the PIO bit.

38.2.3 Interrupt Generation

The QSPI controller generates an interrupt to the processor at the end of a transfer in PIO Mode or when an error is detected (both in PIO and DMA Modes) if the IE.TX or IE.RX bit in the QSPI DMA Control register is enabled for transmit and receive modes of operation, respectively.

If IE.TX is enabled during transmits, an interrupt is generated whenever RDY or one of the FIFO status bits in the QSPI FIFO Control/Status Register is set by the hardware. During transmits, when the QSPI controller is configured as a Slave, if software/APB DMA cannot fill the transmit FIFO fast enough, hardware will set the Tx_FIFO_UNR bit and an underrun condition is generated. If software tries to write to a full Tx FIFO, hardware sets the Tx_FIFO_OVF bit as an indication that software attempted to overflow the Tx FIFO. Hardware makes sure that the overflowing data is never written to the Tx FIFO.

If IE.RX is enabled during receives, an interrupt is generated whenever RDY or one of the status bits in the QSPI FIFO Control/Status Register is set by the hardware. During receives, when the QSPI controller is configured as a Slave, if software/APB DMA cannot read the receive FIFO fast enough, hardware will set the Rx_FIFO_OVF bit and an overflow condition is

generated. However, if software tries to read from an empty Rx FIFO, hardware sets the RXF_UNR bit as an indication that software attempted to underrun the Rx FIFO.

The interrupt can be cleared by writing a 1 to the source of the interrupt. If the interrupt is generated by assertion of RDY, then writing a 1 to the RDY bit clears the interrupt. If the interrupt is generated by assertion of Tx_FIFO_OVF, then writing a 1 to the TXF_OVF bit clears the interrupt. If the interrupt is generated by assertion of Rx_FIFO_UNR, then writing a 1 to RXF_UNR bit clears the interrupt.

38.2.4 Clock Initialization and Control

Refer to Chapter 5: Clock and Reset Controller for details on clock initialization.

38.2.5 Programming Guidelines

This section provides guidelines for programming the QSPI controller:

- Program all the required register fields. Software needs to follow the sequence listed below. Please read this subsection completely to understand the considerations and exceptions):
  a. Clock Mode (Mode 0)
  b. Packed/Unpacked Mode
  c. INTERFACE_WIDTH and SDR_DDR_SEL
  d. Set the bit length (BIT_LEN) and block size (BLOCK_SIZE) values
  e. En_LE_Bit/En_LE_Byte
  f. CS_SW_HW (software based or hardware based)
  - If CS_SW_HW is set, the value on CS_SW_VAL will be driven out.
  - When CS HW is used, program the Setup and Hold Values in the CS Timing Register.
  g. Program the DMA Trig values appropriately, if DMA Mode is used
  h. Enable interrupts
  i. Trimmer settings need to be programmed based on frequency in SDR/DDR modes
  j. Tx_En/Rx_En
  k. Software needs to program num_of_dummy_cycles (required for fast read operation) while setting up a TX transfer (for CMD and ADDR). Note that dummy_cycles are valid only when sw_cs is selected.
  l. For PIO, the QSPI Controller is expected to have all the data in FIFOs before software enables the PIO bit.
  m. PIO or DMA must be programmed last to indicate the start of transfer.

Considerations and Exceptions

PIO or DMA has to be programmed last. These bits indicate the start of transfer. QSPI has sync APB plugin. Because the QSPI registers are in the pclk domain, it takes a couple of additional clock cycles to sync to the qspi_clk domain. In some cases, software might have to add a few clock cycle delay before programming PIO and DMA to make sure that PIO and DMA are programmed last.

For successful operation at various frequency combinations, a minimum 4-5 spi_clk cycle delay might be required before enabling the PIO or DMA bit. This is needed to overcome the MCP between the core and pad macro. The worst case delay calculation can be done considering the slowest qspi_clk as 1 MHz. based on that 1 µs delay should be enough before enabling PIO or DMA.

Once software enables PIO/DMA, no bits are changed until the end of transfer except for PIO/DMA bit. Clearing the PIO/DMA bit will end the transaction. In case the QSPI controller is configured to Receive Mode and software clears the PIO/DMA bit, then the partial data which the controller received until then will be written into the FIFO.

Do not enable conflicting features simultaneously. For example, bidirectional (BIDIR) mode works only with x1 mode.

If any error conditions occur, hardware will set the corresponding status bits in the QSPI FIFO Control/Status register and stop the transfer. If IE is enabled, an interrupt is generated. Software has to clear the source of the interrupt by writing a 1 to it, after the completion of ISR.

DDR output commands are sent to the device in SDR mode, whereas address and data are transferred in DDR mode. So software needs to program the necessary bits along with trimmer settings explicitly.

DDR TX mode uses 2x_clk. The QSPI controller has a clk_en for this clock based on DDR mode selection. When the DDR bit is selected then only the CAR will issue clk_2x to the controller. The CAR block has synchronizer to sync this clk_en. So software might have to wait for a few cycles after selecting DDR mode and before enabling DMA or PIO.

Note that the controller does not generate any interrupts to indicate dummy_cycle done. Software has to wait for some time (based on number of clocks) before setting up the next transfer.

- Set up a TX transfer for CMD and ADDR, configure num_of_dummy_cycles with that
- When the TX transfer is done, wait for some time (based on number of clocks) so that the controller is done with dummy_cycles on the QSPI interface
- Set up an RX transfer to read actual data.

Software needs to make sure that while programming the block_size field, the tx_en and rx_en bits must be zero. This is needed to avoid any internal glitches while loading the block_size value to counters and other comparators. The tx_en and rx_en bits should be programmed separately in the last in COMMAND register.

38.2.5.1 Error Conditions

When the QSPI controller is configured as a Master, the following error scenarios can happen:

- Tx_FIFO overflow

  Tx_FIFO overflow happens when the CPU/APB_DMA writes into the Tx FIFO when it is full. In this case, the QSPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_OVF bit in the QSPI FIFO Control/Status register. The QSPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the QSPI FIFO Control/Status register.

- Rx_FIFO overflow

  Rx_FIFO overflow happens when the CPU/APB_DMA writes into the Rx FIFO when it is full in BOTH_EN Mode. In this case, the QSPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Rx_FIFO_OVF bit in the QSPI FIFO Control/Status register. The QSPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the QSPI FIFO Control/Status register.

- Tx_FIFO underrun

  Tx_FIFO underrun happens when the CPU/APB_DMA reads from the Tx FIFO when it is empty. In this case, the QSPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_UNR bit in the QSPI FIFO Control/Status register. The QSPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_UNR bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the QSPI FIFO Control/Status register.

- Rx_FIFO underrun

  Rx_FIFO underrun happens when the CPU/APB_DMA reads from the Rx FIFO in BOTH_EN Mode. In this case, the QSPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Rx_FIFO_UNR bit in the QSPI FIFO Control/Status register. The QSPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_UNR bit, along with the Interrupt bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the QSPI FIFO Control/Status register.

Note: *In Master Mode, all the errors above can happen only when the CPU/APB_DMA reads/writes an empty/full FIFO. The QSPI controller will never write/read a full/empty FIFO. Instead the*

controller will pause (stops sending clocks to the slave with chip select being in an active state) whenever the FIFOs are full/empty.

38.2.5.2 Programming Trimmers

The 2 programmable trimmers in the QSPI controller are used only in Master Mode:

- QSPI-Tx trimmer

  This trimmer is used in Master Tx mode to adjust/center the outgoing data with respect to the outgoing clock.

- QSPI-Rx trimmer

  This trimmer is used in Master Rx mode to delay the loopback clock to center align the RX data to meet setup/hold and data valid requirements.

In addition, there are 32 tap trimmers on incoming DATA lines as well. These are trimmers can be programmed in the QSPI_TIMING3 register.

38.2.5.3 Tuning Sequence

Tuning is about finding the right sampling point for incoming data. It is required to achieve the trimmer settings which might vary across the platforms. Software needs to follow the tuning procedure below before starting data transfers at higher speeds.

The QSPI controller does not implement auto-tuning. Tuning is completely software-based. Software is supposed to read the tuning pattern and change the trimmer setting accordingly.

However, Spansion flash supports data loss prevention (DLP) where the flash outputs data pattern during dummy cycle phase. Because the controller generated dummy cycles during TX phase, it will not be able to sample DLP data during dummy_cycle phase.

However, software can achieve the DLP scenario by following the steps below.

1. When tuning is not required OR software does not want to read DLP (as in SDR or DDR x1/x2 mode):
   a. Program num_of_dummy_cycle with TX transfer as mentioned above
   b. Set up a normal RX transfer to read main data
2. When tuning is required OR software wants to read/peel DLP:
   a. Programming dummy_cycle is not really required. software can set up a normal RX transfer (length of transfer can be num_of_dummy_cycle + some_extra_bytes), and then peel the DLP data out of it accordingly.
   b. Once tuning is done, software can disable the DLP by writing 0x00 in the VDLP register (using the WVDLR 4Ah command).
   c. Set up the normal RX transfer.

38.2.5.4 Pad Calibration

Software controls the “Reg_on” signal to enable the high-speed interface. This signal is controlled by the QSPI_COMP_PAD_REG_ON bit in the APB_MISC_GP_QSPI_COMP_CONTROL_0 register (see the APB chapter).

This bit has to be set to 0 during cold boot in the boot ROM phase.

In the boot loader phase, this bit should be set to 1. This turns on the regulator inside the comp pad and enables the interface for high-speed operation. Software needs to wait for 1 µs after that. This will turn on the regulator in the pad and the speed can be ramped up to 166 MHz without calibration.

38.3 QSPI Controller Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

There is one set of the QSPI controller registers. The register descriptions below give the offset of each register within the QSPI controller's address range. See Chapter 2: Address Map for the starting address of the QSPI controller.

38.3.1 QSPI_COMMAND_0

QSPI Command1 Register

This register is used to set the bit length and to select the transfer mode.

Chip Select can also be selected to be in hardware mode as software mode with both active high and active low polarities to support devices with varying CS polarities.

Offset: 0x0 | Read/Write: R/W | Reset: 0x4050001f (0b010000xxx10100000xx00x000x011111)

Bit	Reset	Description
31	0x0	PIO: Program 1 after all the other bits in the QSPI_COMMAND2 and QSPI_COMMAND1 registers are programmed to start the transfer. Hardware clears this bit automatically after the transfer is done. Clearing of this bit by software will stop the shifter and latch the partial data into the buffer (in Receive Mode). 0 = STOP (default) 1 = GO
30	0x1	M/S: Master/Slave mode select. 0 = Reserved 1 = Master Mode (internal clock) (default)
29:28	0x0	MODE: The QSPI interface clock mode has to be programmed according to the device with which it is communicating. Only Master Mode 0 is supported. 0 = Mode 0 (default) 1 = RSVD 2 = RSVD 3 = RSVD Master: Only Mode 0 is supported both for Tx and Rx. Slave: Not supported.
27:26	0x0	CS_SEL: Only one chip select is supported to select one slave device. 0 = Selects CS0 (default) 1 = RSVD1 2 = RSVD2 3 = RSVD3
22	0x1	CS_POL_INACTIVE#0: In Master Mode, the inactive value of the external device's CS value, which is connected to CS0, needs to be programmed. 0 = CS0 Inactive value of External device is low 1 = CS0 Inactive value of External device is high 0 = LOW 1 = HIGH (default)
21	0x0	CS_SW_HW: Software control of the QSPI_CS signal in Master Mode. In Slave Mode, this bit need not be programmed. 0 = QSPI _CS# (CS_SEL) is driven to the active state during packet transfers by the hardware (default) 1 = QSPI _CS# (CS_SEL) is driven with the value in the CS_SW_VAL bit 0 = HARDWARE 1 = SOFTWARE
20	0x1	CS_SW_VAL: CS signal value in Master Mode. If CS_SW_HW is 1, then the value in CS_SW_VAL is driven out on CS. If CS_SW_HW is 0, then this bit has no effect. 0 = CS is LOW 1 = CS is HIGH (default)
19:18	0x0	IDLE.SDA: Inactive data signal format. Controls the output enable of the SDA line when the controller is inactive and not doing data transfers. 0 = Drive low. (Master Mode) (default) 1 = Drive high. (Master Mode) 2 = External Pull Down. (Slave Mode) 3 = External Pull high. (Slave Mode)
17	0x0	BIDIR: Bidirectional Transfer Control Bit: This bit enables the bidirectional mode of the transfer. Valid only for SDR mode. 0 = Normal Mode (default) 1 = Bidirectional Mode

Bit	Reset	Description
16	0x0	En_LE_Bit: 1 = Enable Little Endian Bit. 0 = Disable Little Endian Bit 0 = LAST 1 = FIRST
15	0x0	En_LE_Byte: 1 = Enable Little Endian Byte. 0 = Disable Little Endian Byte 0 = LAST 1 = FIRST
12	0x0	Rx_EN: Receive enable. Setting this bit to 1 enables the controller to receive the data. 0 = DISABLE (default) 1 = ENABLE
11	0x0	Tx_EN: Transmit enable. Setting this bit to 1 enables the controller to transmit the data. 0 = DISABLE (default) 1 = ENABLE
9	0x0	SDR_DDR_SEL: Selects between SDR (Single Data Rate) and DDR (Double Data Rate) mode. 0: SDR; Data is transmitted on falling edge of SCLK and received on rising edge of SCLK. 1: DDR; Data is transmitted and received on both the edges of SCLK 0 = SDR 1 = DDR
8:7	0x0	INTERFACE_WIDTH: This field is used to define the QSPI interface width 00 = Single bit mode (x1 mode) 01 = Dual mode (x2 mode) 10 = Quad mode (x4 mode) 11 = RSVD 0 = SINGLE 1 = DUAL 2 = QUAD
5	0x0	PACKED: Packed mode enable bit. 0 = Unpacked Mode (default) 1 = Packed Mode. This is only valid if the BIT_LEN field is set to 7 (8-bit transfer), 15 (16-bit transfer), or 31 (32-bit transfer). When enabled, all 32 bits of data in the FIFO contains valid data packets of either 8-bit, 16-bit, or 32-bit length. Packed mode should be selected when the minimum transfer is 1 word. For bit_len=31, this bit need not be set.
4:0	0x1f	BIT_LEN: This field represents the number of bits in a packet to transmit/receive. The minimum bit_length supported is 1 byte (BIT_LEN=7). Only byte aligned bit lengths are supported (BIT_LEN=7, 15, 31). N = N+1-bit transfer 31 = 32-bit transfer

38.3.2 QSPI_COMMAND2_0

QSPI Command2 Register

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000xx00000000)

Bit	Reset	Description
14:10	0x0	Tx_Clk_TAP_DELAY: Delays the clock going out to the external device with these tap values. Useful only in Master Mode.
7:0	0x0	Rx_Clk_TAP_DELAY: Delays the clock coming in from the external device with these tap values. Useful only in Master Mode.

38.3.3 QSPI_TIMING_REG1_0

QSPI CS Timing1 Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	CS_SETUP_ 0: Specifies the setup time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

Bit	Reset	Description
3:0	0x0	CS_HOLD_0: Specifies the hold time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

38.3.4 QSPI_TIMING_REG2_0

QSPI CS Timing2 Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx100000)

Bit	Reset	Description
5	0x1	CS_ACTIVE_BETWEEN_PACKETS_0: Specifies if CS stays active between two packets on CS#0 1 = CS active between two packets (default) 0 = CS inactive between two packets Note: Hardware-based CS operates only on Flash devices that support it. Please refer to the definition of CS in your Flash device's specification.
4:0	0x0	CYCLES_BETWEEN_PACKETS_0: Specifies the number of cycles between packets in the PIO/DMA Mode for communication on CS#0. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

38.3.5 QSPI_TRANSFER_STATUS_0

QSPI TRANSFER Status Register

Read this register for the status of the transfer.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x0000XXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	RDY: Ready bit. This bit is set to 1 at the end of every transfer, and an interrupt is also generated if the corresponding interrupt enable is set in PIO/DMA Mode. Software writes a 1 to clear it. The interrupt is also cleared when this bit is cleared. 0 = NOT_READY (default) 1 = READY
15:0	RO	X	BLK_COUNT: Counts the number of packets in a transaction (Tx or Rx) in DMA/PIO Mode. When blk_cnt=0xffff is programmed, this field will show 0x0 at the end of a transfer.

38.3.6 QSPI_FIFO_STATUS_0

QSPI Control/Status FIFO Status Register

QSPI STATUS register: Read this register to know the status of the transfer. Error bit is set whenever errors such as Underflow/ Overflow are encountered.

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00400005 (0bxxxxxxxxxxxxxxxx00xxxxx00000xxxx)

Bit	R/W	Reset	Description
29:23	RO	X	RX_FIFO_FULL_COUNT: Indicates the number of slots in the receive FIFO remaining before the FIFO is empty. This field is used by software for debugging purposes.
22:16	RO	X	TX_FIFO_EMPTY_COUNT: Indicates the number of slots in the transmit FIFO remaining before the FIFO is full. This field is used by software for debugging purposes.
15	RW	0x0	RX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Rx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH
14	RW	0x0	TX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Tx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH

Bit	R/W	Reset	Description
13:9	N/A		Reserved for future use. Always write zeros.
8	RW	0x0	ERR: Will be set to 1 by hardware when errors such as underflow/overflow occur. Software writes a 1 to clear the flag. 0 = OK (default) 1 = ERROR
7	RW	0x0	TX_FIFO_OVF: TX FIFO Overflow. This bit is set to 1 whenever the QSPI controller or software tries to write to a full Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TX in the QSPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
6	RW	0x0	TX_FIFO_UNR: TX FIFO Underrun. This bit is set to 1 whenever the QSPI controller or software tries to read from an empty Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TXC in the QSPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
5	RW	0x0	RX_FIFO_OVF: RX FIFO Overflow. This bit is set to 1 whenever the QSPI controller or software tries to write into a full Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the QSPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
4	RW	0x0	RX_FIFO_UNR: RX FIFO Underrun. This bit is set to1 whenever the QSPI controller or software tries to read from an empty Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the QSPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
3	RO	X	TX_FIFO_FULL: TX FIFO Full Status. Hardware sets this bit to 1 if the Tx FIFO is full; otherwise this bit is 0. The Tx FIFO is empty at power on reset (POR). 0 = NOT_FULL (default) 1 = FULL
2	RO	X	TX_FIFO_EMPTY: TX FIFO Empty Status. Hardware sets this bit to 1 if the Tx FIFO is empty; otherwise this bit is 0. The Tx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)
1	RO	X	RX_FIFO_FULL: RX FIFO Full Status. Hardware sets this bit to 1 if the Rx FIFO is full; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_FULL (default) 1 = FULL
0	RO	X	RX_FIFO_EMPTY: RX FIFO Empty Status. Hardware sets this bit to 1 if the Rx FIFO is empty; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)

38.3.7 QSPI_TX_DATA_0

QSPI Transmit Data Register

Offset: 0x18 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	QSPI_Tx_DATA. Transmit Data. This register holds the last data that was transmitted by the QSPI controller.

38.3.8 QSPI_RX_DATA_0

QSPI Receive Data Register

Offset: 0x1c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	QSPI_Rx_DATA. Receive Data. This register holds the last data that was received by the QSPI controller.

38.3.9 QSPI_DMA_CTL_0

QSPI DMA Control Register

QSPI DMA Control Register: Used in the DMA transfers. We can set trigger levels in this register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxx00xx00xxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	DMA_EN: Enable DMA Mode transfer. Software writes a 1 to this bit to start a transfer in the DMA mode. All fields in the QSPI_COMMAND1 and QSPI_DMA_CTL register must be set before writing a 1 to this bit. This bit is cleared by the QSPI controller after all packets have been transferred as indicated by the DMA_BLOCK_SIZE field. Clearing this bit by software will stop the shifter and latch the partial data into buffer. 0 = DMA Mode is disabled (default) 1 = DMA Mode is enabled
20:19	0x0	RX_TRIG: Receive FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is at least 1 packet in the RX FIFO. (default) 01: 4 words. DMA trigger is asserted when there are at least 4 packets in the RX FIFO. 10: 8 words. DMA trigger is asserted when there are at least 8 packets in the RX FIFO. 11: 16 words. DMA trigger is asserted when there are at least 16 packets in the RX FIFO. (Presently, the APB DMA does not support this trigger level.)
16:15	0x0	TX_TRIG: Transmit FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is space for at least 1 packet in the TX FIFO. (default) 01: 4 words. DMA trigger is asserted when there is space for at least 4 packets in the TX FIFO. 10: 8 words. DMA trigger is asserted when there is space for at least 8 packets in the TX FIFO. 11: 16 words. DMA trigger is asserted when there is space for at least 16 packets in the TX FIFO. (Presently, the APB DMA does not support this trigger level.)

38.3.10 QSPI_DMA_BLK_SIZE_0

QSPI DMA Block Size Register

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DMA_BLOCK_SIZE: Size of data block to be transferred in PIO/DMA Mode. The default is 0. In DMA Mode, the maximum number of 32-bit packets that can be transferred is 2^{16} = 65536. In PIO Mode, the maximum number of 32-bit packets that can be transferred is 64 (FIFO depth). The Block Size has to be programmed 1 packet less than intended; for example, a value of 3 indicates 4 packets are to be transferred.

38.3.11 QSPI_TX_FIFO_0

QSPI TX FIFO Buffer Register

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	QSPI Tx_FIFO_REGISTER. Tx FIFO.

38.3.12 QSPI_RX_FIFO_0

QSPI RX FIFO Buffer Register

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	QSPI Rx_FIFO_REGISTER. Rx FIFO.

38.3.13 QSPI_INTR_MASK_0

QSPI Interrupt Mask Register

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxx00000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	0x0	RDY_INTR_MASK: Interrupt enable mask bit for RDY. 0 = Enable interrupt generation when RDY bit is asserted. 1 = Disable interrupt generation when RDY bit is asserted. 0 = DISABLE 1 = ENABLE
28	0x0	TX_FIFO_OVF_INTR_MASK: Interrupt enable mask bit for TX_FIFO_OVF. 0 = Enable interrupt generation when TX_FIFO_OVF is asserted. 1 = Disable interrupt generation when TX_FIFO_OVF is asserted. 0 = DISABLE 1 = ENABLE
27	0x0	TX_FIFO_UNF_INTR_MASK: Interrupt enable mask bit for TX_FIFO_UNF. 0 = Enable interrupt generation when TX_FIFO_UNF is asserted. 1 = Disable interrupt generation when TX_FIFO_UNF is asserted. 0 = DISABLE 1 = ENABLE
26	0x0	RX_FIFO_OVF_INTR_MASK: Interrupt enable mask bit for RX_FIFO_OVF. 0 = Enable interrupt generation when RX_FIFO_OVF is asserted. 1 = Disable interrupt generation when RX_FIFO_OVF is asserted. 0 = DISABLE 1 = ENABLE
25	0x0	RX_FIFO_UNF_INTR_MASK: Interrupt enable mask bit for RX_FIFO_UNF. 0 = Enable interrupt generation when RX_FIFO_UNF is asserted. 1 = Disable interrupt generation when RX_FIFO_UNF is asserted. 0 = DISABLE 1 = ENABLE

38.3.14 QSPI_SPARE_CTLR

QSPI Spare Control Register

Offset: 0x190 | Read/Write: R/W | Reset: 0x0fff0000 (0b00001111111111111000000000000000)

Bit	Reset	Description
31:11	00001111111111100000b	Reserved for future use.
10:8	000b	QSPI_SPARE_CONTROL_REGISTER_BYTE2. Program bits 8, 9, and 10 along with Rx_Clk_TAP_DELAY in the COMMAND2 register to adjust the clock delay on internal registers. Useful in Master Mode only.

Bit	Reset	Description
7:0	0x00	Reserved for future use.

38.3.15 QSPI_MISC_0

QSPI Miscellaneous Register

Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	NUM_OF_DUMMY_CLK_CYCLES: Number of dummy cycles required in case of Fast read commands. this is useful only in case of read from flash.

38.3.16 QSPI_TIMING3_0

QSPI Timing3 Register

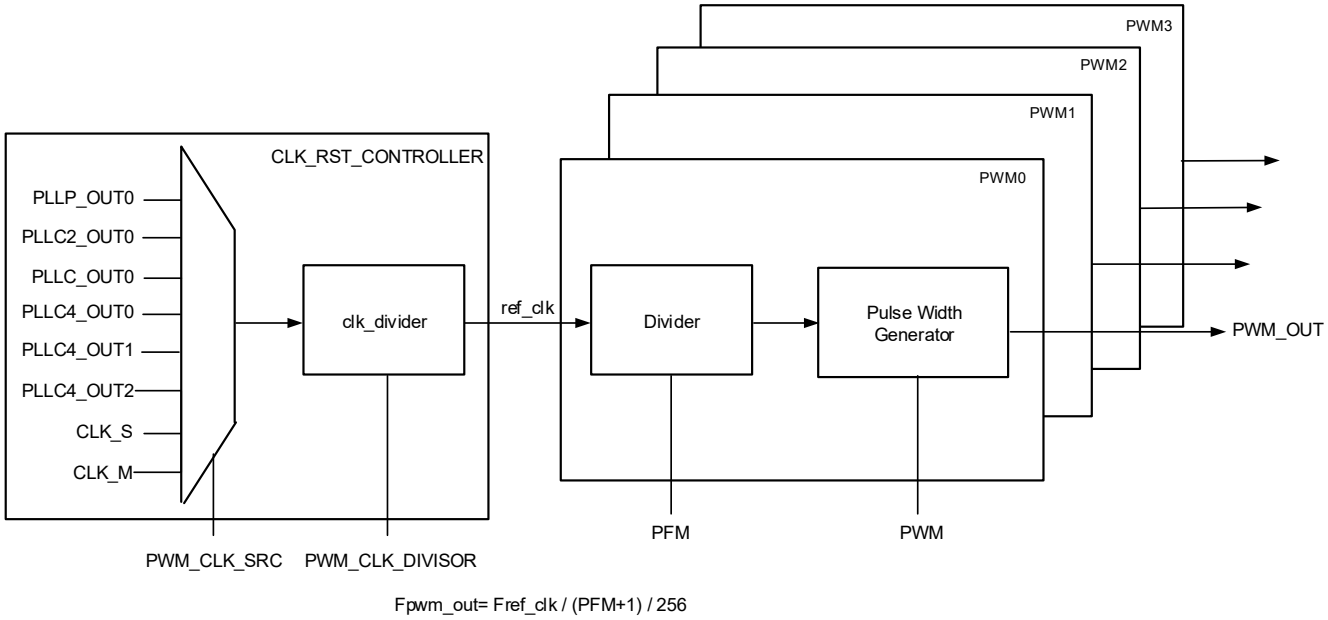
Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxx00000xxx00000xxx00000xxx00000)

Bit	Reset	Description
28:24	0x0	DATA3_LINE_TAP_DELAY: Delays the Data3 coming in from the external device with these tap values.
20:16	0x0	DATA2_LINE_TAP_DELAY: Delays the Data2 coming in from the external device with these tap values.
12:8	0x0	DATA1_LINE_TAP_DELAY: Delays the Data1 coming in from the external device with these tap values.
4:0	0x0	DATA0_LINE_TAP_DELAY: Delays the Data0 coming in from the external device with these tap values.

CHAPTER 39: PWM CONTROLLER

The Pulse Width Modulator (PWM) controller is a four channel frequency divider whose pulse width varies. Each channel has a programmable frequency divider and a programmable pulse width generator.

Figure 166: PWM Controller Block Diagram



Note: *To derive PFM given the Fref_clk and desired Fpwm_out:*
PFM = max(min(round((Fref_clk / 256)/Fpwm_out)-1, 0x1fff), 0x0)
Where round() rounds result to nearest integer and min/max clamp the PFM to the valid 13-bit range.
Also, PWM computation: Given desired pulse with as a percentage (0..100 %):
*PWM = max(min(round(percent/100 * 256), 256), 0)*

Frequency division is a 13-bit programmable value, and pulse division is an 8-bit value.

The PWM controller runs off a device clock, which is programmed in the Clock and Reset controller, and can be any frequency up to the device clock maximum speed of 48 MHz.

The device clock frequency is always subject to a minimum divide by 256 to generate the PWM output frequency, and may also be subdivided to slow it down further based on the programmable value locally.

There is an APB interface that interfaces the register logic to the APB bus.

The PWM controller contains four independently programmable pulse width modulators. Each generated pulse has an n/256 duty cycle. PWM signals are useful for LCD contrast and brightness control, VCO-generated clocks and other analog voltage references where high precision is not required.

39.1 Functionality

There are four PWM controllers on four individual pins on the chip. The pinmux corresponding to these are SDC primary pin groups. For the four PWM controllers, there are four corresponding registers:

- PWM_CSR0
- PWM_CSR1

- PWM_CSR2
- PWM_CSR3

Each PWM controller must be enabled (bit 31) to be operational. Pulse Width [24:16] determines the output pulse width and must be programmed appropriately. Pulse Width [30:25] is not used and must be 0. Frequency Divider [12:0] determines the divided clock.

39.2 PWM Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

39.2.1 PWM_CONTROLLER_PWM_CSR_<x>_0

There are four PWM CSR registers, where <x> is a value from 0 through 3.

PWM Output <x> Configuration Control Register

Offset: 0x0 + (<x> * 0x10) | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable Pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed. 0=Always low. 1=1/256 Pulse high. 2=2/256 Pulse high. N=N/256 Pulse high Note: To achieve 100% duty cycle (constant high output), program register PWM_CONTROLLER_PWM_CSR_0 Bit 24 to 1.
12:0	0x0	PFM_<x>: Frequency divider that needs to be programmed. See the note on programming the PFM value at the start of this chapter.

CHAPTER 40: THERMAL SENSOR AND THERMAL THROTTLING CONTROLLER

40.1 Thermal Throttling Controller (SOC_THERM)

The functions of the thermal throttling controller (SOC_THERM) include:

- Thermal Sensor Management: Handles access, capture and processing of data from thermal sensors which are located in multiple locations around the SOC.
- Thermal Event Detection: Provides multiple software configurable thermal thresholds per sensor. Thermal threshold crossings can be configured to raise interrupts or trigger a hardware throttling response.
- Over-Current Detection:
  - Externally Signaled Event Detection: Configurable “over-current” (OC) input pins which can be used to trigger throttle responses. These pins might be wired to signals from the PMIC or battery monitors.
- Throttle Management and Prioritization: For each of the events that can trigger a throttle response, provides configuration of that throttling response. For example, a CPU temperature exceeding 90ºC could trigger a 50% throttling of the CPU clock. Also, this block provides for prioritizing the throttling responses when multiple thermal or over-current events are happening simultaneously.

The intention of this section is not to provide a full programmer’s guide, but solely to document the register interface to allow better understanding of NVIDIA provided drivers.

40.2 Thermal Sensor

Thermal sensors are used to constantly monitor the temperature on the chip. Sensors are placed across the die to gauge the temperature of the whole chip. The TSensor module can:

- Generate an interrupt to software to lower temperature via DVFS, on reaching a certain thermal threshold
- Generate a signal to the CAR block to reduce CPU frequency by half, on reaching a certain thermal threshold
- Generate a signal to the PMC when temperature reaches very high levels to reset the chip, and set a flag in the PMC

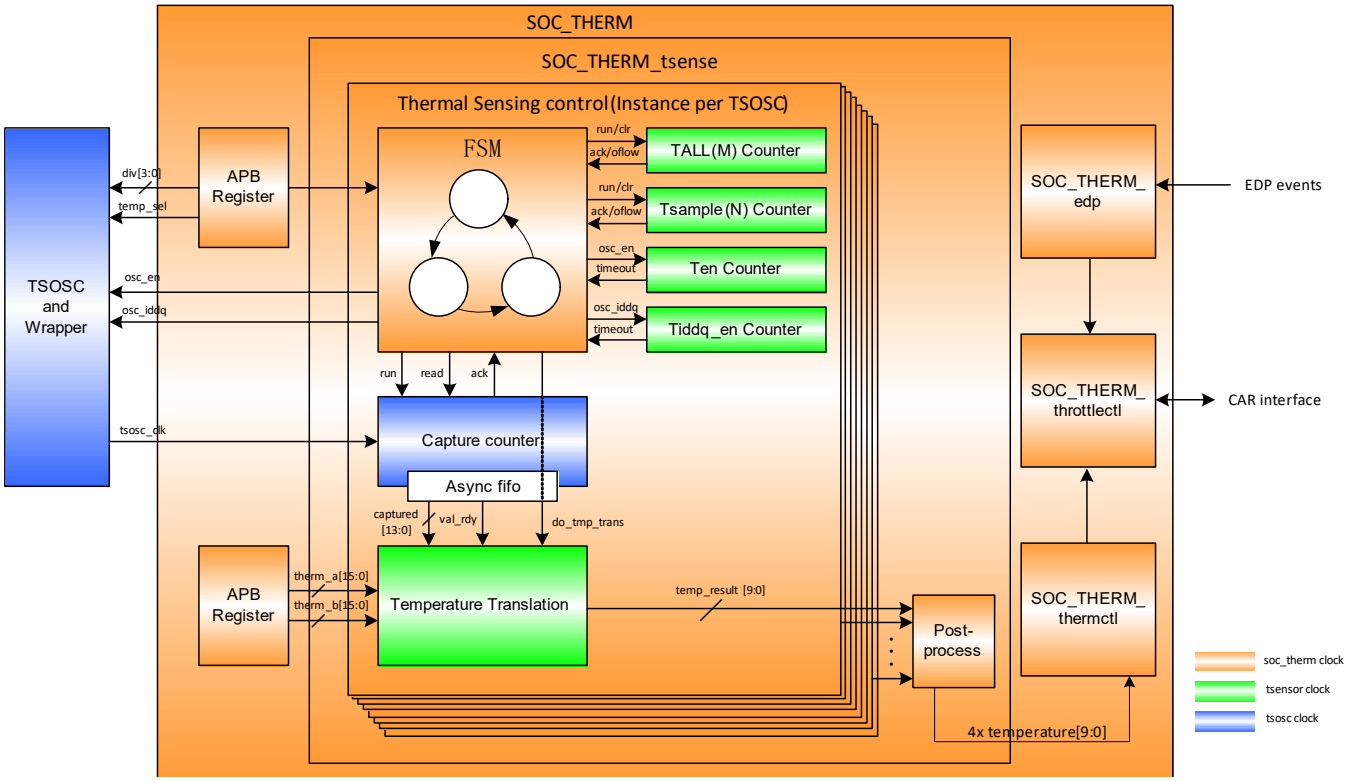
40.3 Thermal Sensing (SOC_THERM_tsense)

SOC_THERM_tsense centralizes management of eight on-chip thermal sensors: one per CPU core, one within the GPU, two near the memory I/O, and one near PLLX. Each sensor is an instance of the TSOSC_C cell -- a ring oscillator whose output frequency is proportional to temperature.

SOC_THERM_tsense includes

- capture logic for manipulating the TSOSC’s and measuring their output
- arithmetic logic for translating sensor readings to degrees Celsius (with 0.5ºC precision)
- post-processing logic (e.g., for handling “invalid” sensor data) and
- JTAG interface logic to support per-sensor calibration on ATE.

Figure 167: SOC_THERM_tsense in context



The output of SOC_THERM_tsense is a stream of four temperature measurements – CPU temperature, GPU temperature, memory I/O temperature, and PLLX temperature. SOC_THERM_tsense feeds these measurements directly to SOC_THERM_thermalctl but also exposes them to software via registers.

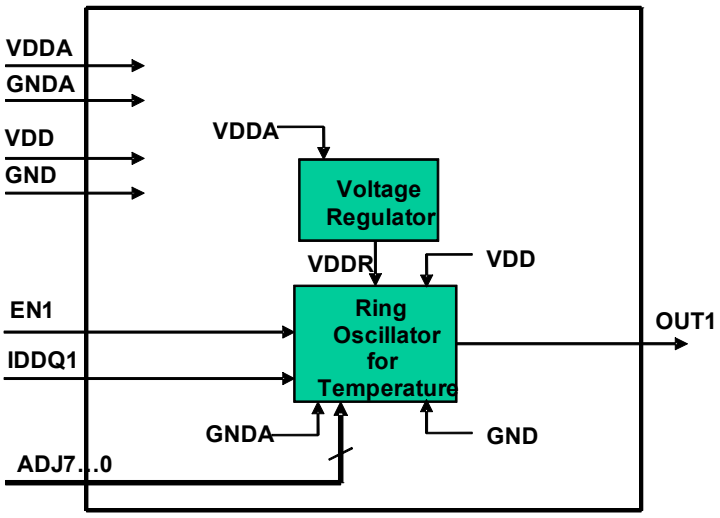
Temperature translation depends on per-sensor calibration data. This calibration data is collected on ATE (during chip manufacture) and stored in on-chip fuses. Prior to enabling thermal sensing, software must unpack the calibration data and pass it to SOC_THERM_tsense's translation logic via registers.

The real TSOSC Vmin will be higher than real standard logic Vmin. There is a software scheme to disable VDD TSOSCs when voltage goes too low and hardware scheme to redirect readings to I/O TSOSC with hot spot displacement. It is possible to overheat at Vmin.

40.3.1 TSOSC Thermal Sensor Cells

The figure below shows the basic TSOSC structure.

Figure 168: TSOSC Structure



The output frequency of any CMOS ring oscillator depends on manufactured characteristics (i.e., process), supply voltage, and temperature. When enabled at steady state, TSOSC acts as a temperature sensor because:

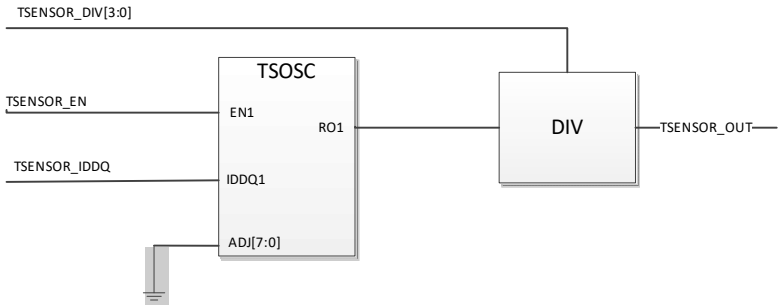
- the voltage regulator compensates for variation in supply voltage (VDDA/GNDA) and
- calibrate every TSOSC we manufactured is calibrated to compensate for manufacturing variation

Logic within SOC_THERM_tsense translates TSOSC output frequency to temperature. The TSOSC datasheet includes a table estimating TSOSC output frequency relative to temperature and silicon process. The content of that table is critical for validating the precision required in the SOC_THERM_tsense arithmetic.

40.3.1.1 Wrapper Logic

In order to decrease number of wires running across the chip and reduce RO frequency to the value safe to pass for a long distances each TSOSC is placed within a simple wrapper.

Figure 169: TSOSC Wrapper



SOC_THERM_tsense is on VDD_CORE. Therefore, signals between SOC_THERM_tsense and the CPU and GPU thermal sensors pass through level shifters and clamps.

To facilitate temperature processing in the capture unit Power Valid signals for each CPU and for the GPU should be generated. They should be deasserted when corresponding units are power gated and for CPUs – all 4 signals should be deasserted when VDD_CPU is rail gated.

When Tegra comes out of reset IDDQ signals will be asserted to all thermal sensors to minimize the leakage current.

TSENSOR_OUT must be <= ~50 MHz to facilitate its routing to the capture logic within SOC_THERM_tsense.

40.3.1.2 Placing and Powering TSOSCs (and Wrappers)

Tegra X1 has 9 TSOSCs powered as follows:

Table 241: Thermal Sensors in Tegra X1

Sensor Name	Sensor Type	VDD Connection	VDDA Connection	Comment
TS_AO	TSOSC_HV	VDD_RTC	VDDIO_SYS (1.8V)	Co-locate with TDIODE in pad-ring This TSOSC is controlled via PMC, not SOC_THERM. See the PMC chapter.
TS_PLLX	TSOSC_C	VDD_CORE	AVDD_PLLX	Placed adjacent to PLLX to shorten power trace from AVDD_PLLX and use its ESD protection
TS_GPU	TSOSC_C	VDD_GPU	VDD_GPU	Locate equidistance from the 2 SMs and as close as possible.
TS_MEM0	TSOSC_C	VDD_CORE	VDD_CORE	Placed between data bricks of pad blocks of memory channels 0 and 1 accordingly.
TS_MEM1	TSOSC_C	VDD_CORE	VDD_CORE	
TS_CPU0	TSOSC_C	VDD_CPU	VDD_CPU	Ideally, it is centrally located in the CPU. However, practical issues forced final location is on the outer edge.
TS_CPU1	TSOSC_C	VDD_CPU	VDD_CPU	
TS_CPU2	TSOSC_C	VDD_CPU	VDD_CPU	
TS_CPU3	TSOSC_C	VDD_CPU	VDD_CPU	

TSOSC requires two voltages digital (VDD) and analog (VDDA). VDDA can be connected either to I/O power through direct drill attachment or to core power. It doesn't have built-in ESD devices, so if connected to I/O power through direct drill it should be placed as near as possible to device having ESD protection.

40.3.1.3 TSOSC Vmin Limitations

Because TSOSCs include a built in LDO, they can provide accurate temperature values over a wide range of voltages. However, they do have a minimum operation voltage, below which sensor accuracy drops precipitously. Because of the LDO, this minimum operating voltage is consistently higher than the logic Vmin.

Managing TSOSC Minimum Voltage Requirements

SOC_THERM anticipated this problem and has a built in mitigation strategy to fall back to the PLLX-TSOSC when voltage is too low for the core-voltage TSOSCs to function. PLLX-TSOSC is powered on the AVDD_PLLX rail and does not suffer from Vmin concerns, but is further from the hotspot than the sensors it replaces.

There are a number of conditions that cause TSOSC voltage to be too low for the TSOSCs to function. Fallback to PLLX-TSOSC happens when hardware or software indicates a TSOSC is invalid. Ultimately, **software is responsible** for disabling a given thermal sensor before its VDDA drops too low. The following table lists the scenarios where TSOSC enable/disable awareness is required and which software agents are responsible.

Table 242: Agents Responsible for TSOSC Vmin Management

TSOSC Scenario	Agent Responsible	Using Which Register
CPU{0..3}, on-rail gating change	BPMP-L	TSENSOR_VALID
CPUn, on power-gating	Unnecessary	Unnecessary
CPU{0..3}, on voltage change	Kernel DVFS	SENSOR_STOP
CPU{0..3}, CC3	PMC	PMC2SOC_THERM_CRAIL_VOLTAGE_VALID
MEM{0,1}, on VDD_CPU voltage change	Kernel DVFS	SENSOR_STOP
GPU0, on voltage change	Kernel DVFS	SENSOR_STOP
GPU0, on rail-gating change		

PLLX_TSOSC Fall-Back Procedure

The PLLX_TSOSC is further away from the hotspot than the core TSOSCs. So the thermal sensing hardware adds a programmable offset to sensed temperatures to compensate for this.

To prevent temperature underestimation, this offset should be programmed to the difference between the hotspot deltas of PLLX_TSOSC and CPU_TSOSC (or other sensor group TSOSCs). However, programming in this way can cause

discontinuities in sensed temperature when the actual hotspot-delta is less than the worst-case delta. These discontinuities can cause immediate hardware shutdown or abrupt thermal governing transients.

To avoid this, the procedures below program the offset to be the actual measured difference between PLLX_TSOSC and CPU_TSOSC, immediately before switching to PLLX_TSOSC. Doing so makes temperatures smooth around the transition to PLLX_TSOSC, but at the risk of temperature underestimate if temperature gradient increases. This is an acceptable risk because gradients are expected to go DOWN when at Vmin. There is still a discontinuity on switch BACK to CPU TSOSC, but the expectation is that this will be tolerably small.

TSOSCs on VDD_CPU

If decreasing CPU frequency to below a predetermined threshold (Fmax@Vmin_tsosc):

1. Read PLLX and CPU_TSOSC temperature values.
2. Adjust CPU_HOTSPOT_OFF by MAX(CPU_TSOSCs) – PLLX.
3. Disable CPU_TSOSCs using SENSOR_STOP field in associated TSENSOR_SENSOR(N)_CONFIG registers.
4. Program VDD_CPU voltage target.

If increasing CPU frequency to the same predetermined threshold, or greater:

1. Program VDD_CPU voltage target.
2. Wait till voltage change is known to have happened. (SOC_THERM driver does not know when voltage change request completes. One solution is to use a timer set to a high-confidence maximum I2C delay.)
3. Enable CPU_TSOSCs using SENSOR_STOP.
4. Restore CPU_HOTSPOT_OFF to its default value.

The same procedure applies to sensors on VDD_GPU or VDD_CORE.

40.4 Temperature Translation Logic

SOC_THERM_tsense contains one instance of “temperature translation” logic per TSOSC. This logic takes a 16-bit count from the capture logic and performs a linear scaling (that is, Ax+B) to convert to temperature in degrees C.

```
wire signed [SOC_THERM_TSENSOR_CPU0_CONFIG2_0_THERM_A_SIZE-1:0] therm_a;
wire signed [SOC_THERM_TSENSOR_CPU0_CONFIG2_0_THERM_B_SIZE-1:0] therm_b;
reg  signed [29:0] temp_product;
reg  signed [17:0] temp_sum;
&Vector 14 captured;
&Vector 10 temperature_translated;

// therm_a * captured + therm_b
temp_product = therm_a * $signed({1'b0, captured});
// truncate lower 13 bits before adding with therm_b
temp_sum = $signed(temp_product[29:13]) + therm_b;
// overflow/underflow if all sign bits are not equal
overflow_underflow = !(&temp_sum[17:9]) && (|temp_sum[17:9]);
temperature_translated <0= temp_sum[9:0];
```

The resulting temperature is 10-bit signed and goes to throttling logic as a 10-bit signed. Coefficients will be programmed to get 1/2ºC units. To make reading back temperatures easier for humans – sign and bit 0 are split into separate readback bytes, so the high byte will just read absolute value in ºC. Lower byte will keep LSB in bit 7 and sign in bit 0. Bit assignment of 16-bit readback register will be:

- 15:8 – temperature absolute value high bits (with 1/2ºC programming – it will be the temperature in ºC

- 7 – LSB. 1/2ºC bit.
- 0 – sign bit. Negative temperatures will not be very readable, but likely, not many will be measured.

40.5 Post-Processing Logic

The capture and translation logic does its best to produce a temperature reading for each TSOSC. However, sensor readings may not be available due to a stopped (e.g., powered-down) sensor or due to overflow. The post-processing logic takes all available measurements, all error indications, and synthesizes four final temperature values – one each for CPU, GPU, MEM I/O and PLLX. The post-processing logic forwards these final temperature values to SOC_THERM_thermctl.

40.5.1 Calibration Fusing and ATE

The TSOSCs (and capture and translation logic) are only useful as thermal sensors if they are properly calibrated to compensate for process variation. So, as part of chip manufacture and test (ATE), a calibration process is executed. The TSOSC output frequency is measured at two temperatures and the result is fused into the chip. Then, at runtime, software reads those fused values and programs the temperature translation logic accordingly.

Tsosc hardware fuse map is mapped to the functional fields as follows:

Table 243: TSOSC Related Fuses in Tegra X1

Fuse Name	Logical Bits	Physical Fuses	Offset	Purpose	Functional fields
tsensor_common	32	64	0x280	CP1 and CP2 temperature and offset	temp_shift_cp[5:0], temp_shift_ft[4:0], base_calib_cp[9:0], base_calib_ft[10:0]
tsensor0_calib	26	52	0x198	CPU0-TSOSC at CP1 and CP2	tsensor0_calib_cp1[12:0], tsensor0_calib_cp2[12:0]
tsensor1_calib	26	52	0x184	CPU1-TSOSC at CP1 and CP2	tsensor1_calib_cp1[12:0], tsensor1_calib_cp2[12:0]
tsensor2_calib	26	52	0x188	CPU2-TSOSC at CP1 and CP2	tsensor2_calib_cp1[12:0], tsensor2_calib_cp2[12:0]
tsensor3_calib	26	52	0x22C	CPU3-TSOSC at CP1 and CP2	tsensor3_calib_cp1[12:0], tsensor3_calib_cp2[12:0]
tsensor4_calib	26	52	0x254	MEM0-TSOSC at CP1 and CP2	tsensor4_calib_cp1[12:0], tsensor4_calib_cp2[12:0]
tsensor5_calib	26	52	0x258	MEM1-TSOSC at CP and CP2	tsensor5_calib_cp1[12:0], tsensor5_calib_cp2[12:0]
tsensor6_calib	26	52	0x25C	GPU-TSOSC at CP1 and CP2	tsensor6_calib_cp1[12:0], tsensor6_calib_cp2[12:0]
tsensor7_calib	26	52	0x260	PLLX-TSOSC at CP1 and CP2	tsensor7_calib_cp1[12:0], tsensor7_calib_cp2[12:0]
tsensor9_calib	26	52	0x2D4	AOTAG TSOSC at CP1 and CP2	tsensor9_calib_cp1[12:0], tsensor9_calib_cp2[12:0]
tsensor10_calib	26	52	0x31C	Two TSOSCs TBD, at EXT	Tsensor10a_calib_ext[12:0], Tsensor10b_calib_ext[12:0]
tsensor10_calib_aux	16	32	0x320	EXT*(extended char)* temperature and offset	temp_shift_ext[5:0], base_calib_ext[9:0],

TSOSC calibration is done at two temperatures, called CP1 and CP2 (also known as FT). Target temperatures are CP1=25°C and CP2=100°C respectively, but real temperature may be +/-12°C for CP and +/-5ºC for CP2. So real temperature (from TjADT) is burned into the fuses. To save the bits the temperature shift is burned (in ½ ºC units) relative to target 25°C and 100°C temperatures. The real temperatures (in 1/2ºC units) at the time of measurement are reconstructed from fuse values as follows:

```
int32 capture_temp_cp1_2X = 25<<1 + ((int32)(temp_shift_cp<<(32-6)))>>(32-6);
int32 capture_temp_cp2_2X = 100<<1 + ((int32)(temp_shift_ft<<(32-5)))>>(32-5);
```

TSOSC frequency is measured as the number of oscillator periods counted during some fixed period of time. Calibration measurements are compressed in a (common) average plus per-TSOSC displacement to save on the number of fuse bits. The counts captured at CP1 (tsensorN_capture_cp1[15:0]) and at CP2 (tsensorN_capture_cp2) are reconstructed from fuse values as follows:

```
int32 tsensorN_capture_cp1 = base_calib_cp<<6 + ((int32)(tsensorN_calib_cp1<<(32-13)))>>(32-13);
int32 tsensorN_capture_cp2 = base_calib_ft<<5 + ((int32)(tsensorN_calib_cp1<<(32-13)))>>(32-13);
```

40.5.2 Calculating Therm_A and Therm_B from Fuse Values

The formulae to calculate THERM_A and THERM_B are:

$$\text{Therm\_A} = \frac{\text{CP2\_fuse\_temp\_in\_1/2\_C} - \text{CP1\_fuse\_temp\_in\_1/2\_C}}{\text{CP2\_fuse\_capture} - \text{CP1\_fuse\_capture}} \cdot 2^{13} \cdot \frac{\text{PDIV}}{\text{PDIV}_{\text{ATE}}} \cdot \frac{F_{\text{tsensor}}}{F_{\text{tsensor,ATE}}} \cdot \frac{\text{TSAMPLE}_{\text{ATE}}}{\text{TSAMPLE}}$$

$$\frac{\text{CP2\_fuse\_capture} \cdot \text{CP1\_fuse\_temp\_in\_1/2\_C} - \text{CP1\_fuse\_capture} \cdot \text{CP2\_fuse\_temp\_in\_1/2\_C}}{\text{CP2\_fuse\_capture} - \text{CP1\_fuse\_capture}}$$

The extra ratios in the THERM_A equation are to adjust for difference in capture settings between ATE calibration using JTAG and actual operation using soc_therm. ATE settings and suggested SOC_THERM settings are given in the table below. Note that programmed TSAMPLE values should be one less than value given in the table.

Table 244: Coefficients for ThermA/B Calculations in Tegra X1

	All-TSOSCs except AOTAG	AOTAG
PDIV	8	8
PDIV_{ATE}	8	8
F_{tensor}	400 kHz	
$F_{\text{tensor,ATE}}$	400 kHz	
TSAMPLE	120	
$\text{TSAMPLE}_{\text{ATE}}$	479	

40.5.3 Software Override of SOC_THERM_tsense Output

For testing SOC_THERM and related software drivers, it can be useful for software to be able to spoof the output of SOC_THERM_tsense. The _TEMP_SW_OVERRIDE fields and the _TEMP fields provide a mechanism for doing just that.

This override technique has limitations. When software override mode is used, it is expected that software periodically updates temperature registers, similar to the hardware sensor updates. When threshold levels are changed, it is important to follow this up with a temperature update from software so that internal threshold checks behave correctly.

40.6 Thermal Event Detection (SOC_THERM_thermctl)

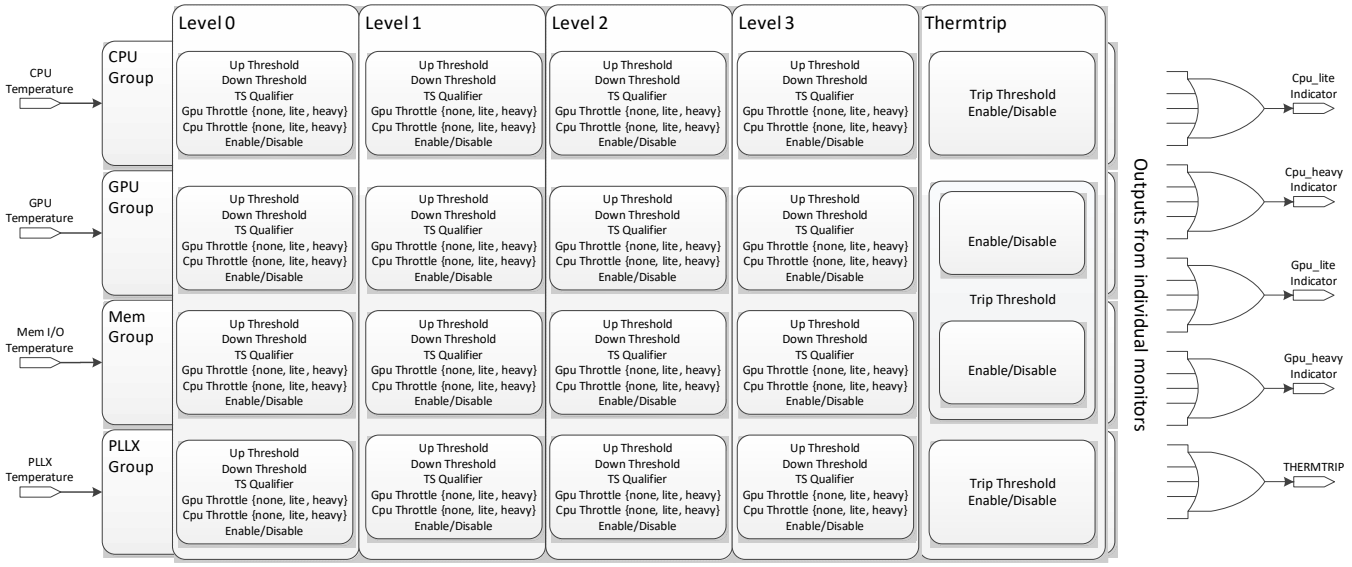
SOC_THERM_thermctl monitors temperatures coming from SOC_THERM_tsense and takes software-configured actions at software-configured temperature thresholds. The available actions are:

- generating an interrupt
- asserting one or more “throttle indicators” to SOC_THERM_throttlectl [1]or
- invoking “thermtrip” to shut down the system.

1. Depending on configuration of throttlectl, an asserted throttle indicator may lead to throttling of the GPU or CPU clock.

SOC_THERM_thermctl is organized as a two dimensional array – groups (horizontally) and levels (vertically). There are four groups – one each for CPU temperature, GPU temperature, memory I/O temperature, and PLLX temperature. There are 4 generic levels and one thermtrip level. Each intersection of a group and level has a “monitor”. Each monitor includes its own thermal threshold and its own programming of actions.

Figure 170: SOC_THERM_thermalctl Block Diagram



Each level is independent from the others. The groups are largely independent of one another with a few notable exceptions:

- Thermtrip for GPU temperature and memory I/O temperature share a trip threshold
- The behavior of any monitor with its “TS qualifier” bit set depends on the state of the PLLX temperature monitor within the same level.

Each monitor outputs a few “throttle request” signals – one each for “gpu_lite”, “gpu_heavy”, “cpu_lite” and “cpu_heavy”. SOC_THERM_thermctl combines these signals from the many monitors to produce a single set of “gpu_lite, “gpu_heavy”, “cpu_lite”, and “cpu_heavy” throttle indicators which it forwards to SOC_THERM_throttlectl.

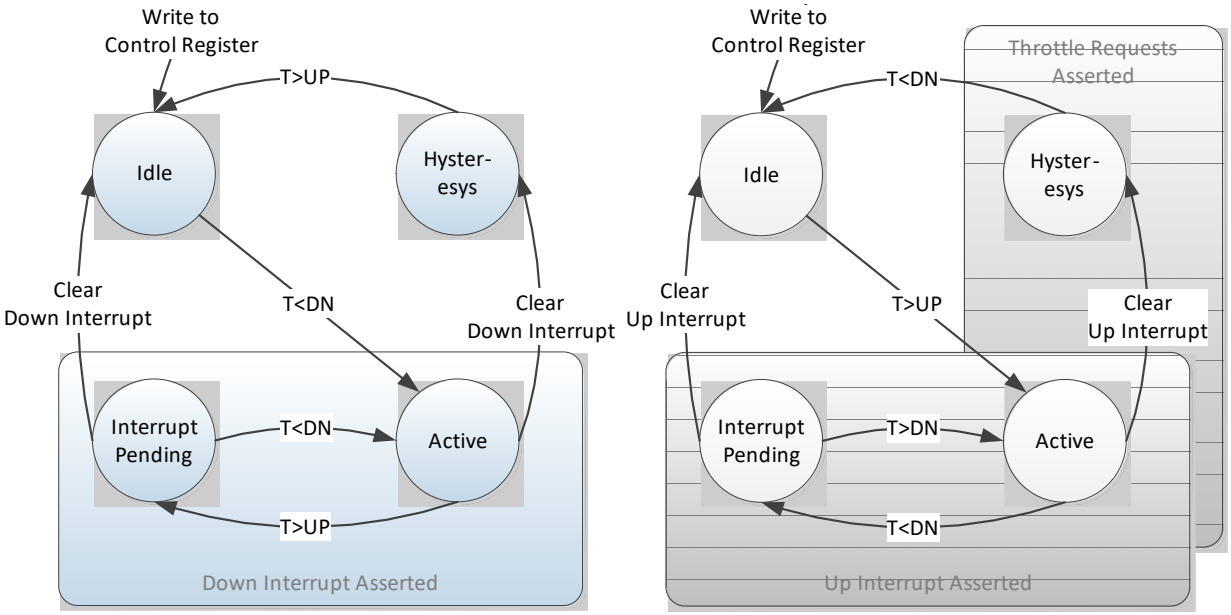
40.6.1 Monitors for Levels 0-3

Levels 0 – 3 are completely identical and interchangeable. Within a given level, there is one monitor per sensor group. The purpose of a monitor is to detect particular thermal conditions and take software-configured actions. Each monitor has several configuration options:

- EN – should the monitor operate or not?
- UP – A temperature threshold. Temperature rising beyond this threshold causes the monitor to assert its throttle requests
- DN – A temperature threshold. Temperature falling below this threshold causes the monitor to deassert its throttle requests.
- GPU – what GPU throttle request to issue. Can be “none”, “lite”, or “heavy”
- CPU – what CPU throttle request to issue. Can be “none”, “lite”, or “heavy”
- INTR_UP – should the monitor raise an interrupt when temperature rises above UP?
- INTR_DN -- should the monitor raise an interrupt when temperature falls below DN?
- TS – should the monitor operate on its own or should its behavior depend on the PLLX monitor at the same level?

A monitor includes two separate FSM's for generating the up and down interrupts. The state machines implement hysteresis. So, for example, once a "down interrupt" has been asserted and cleared it won't be re-asserted until after temperature has risen above UP. Similarly, once an "up interrupt" has been asserted and cleared it won't be re-asserted until after temperature has fallen below DN. Software can force a transition back to the idle by writing to the monitor's control register (SOC_THERM_THERMCTL_LEVEL<N>_GROUP_<S>).

Figure 171: Monitor State Machines (Ignoring TS)



Conceptually, a monitor asserts its throttle requests based on the state of the FSM which generates the up interrupt. As a consequence, the monitor cannot request "cold throttling" (i.e., throttle below a particular temperature). If "cold throttling" is required it must be implemented in software.

40.6.1.1 TS Qualifier Bit

Each monitor has a TS qualifier bit in its control register. When the bit is clear, the monitor operates as described above. When the bit is set, the monitor state machine transitions depend on the PLLX monitor at the same level. For example, suppose the following conditions are set:

- CPU_UP_THRESHOLD for level = 1 is set to 75ºC
- TS = 1 for SOC_THERM_THERMCTL_LEVEL1_GROUP_CPU
- TSENSE_UP_THRESHOLD for level = 1 is set to 70ºC

When the CPU_UPTHRESHOLD is breached (i.e., CPU sensor reports temperature > 75ºC), INTR_UP for CPU group is raised only if TSENSE reports temperature > 70ºC (i.e., the TSENSE_UP_THRESHOLD for Level=1 is also breached). The assumption is that TSENSE is more accurate than other sensors and this qualification can help reduce the frequency at which CPU, GPU, MEM is throttled based on their own tsense readouts. The same holds for breaching down thresholds.

40.6.1.2 Power Valid

Power-Good Signal from PMC

PMIC provides a power-good signal which is multiplexed with the PMIC over-current alarm. This is routed to the PMC which asserts a level signal to SOC_THERM when (and as long as) VDD_CPU power is good. Throttling actions for the CCPLEX are disabled when CPU pwrgood signal is de-asserted. The throttle control is applied in the SOC_THERM_throttlectl block.

40.6.1.3 Spurious Interrupts

At Startup

At start-up, spurious DN interrupts may be generated if DN thresholds are not programmed correctly. Software can prevent this by either:

1. Disabling the DN interrupt until the first UP threshold is breached.
2. Program the Level0 DN interrupt to a very small value (e.g., -50ºC), so that Level0 DN interrupt is effectively disabled. When temperature breaches the first Level0 UP threshold, software can program the DN threshold to an appropriate value to define a thermal zone.

Note: *No special hardware support is required that establishes any dependency between UP and DN thresholds/interrupts for a particular level. Creation of thermal zones is software's responsibility – hardware only provides indication of temperature moving outside the current zone.*

At Reprogramming

As mentioned previously, software is expected to define thermal zones by re-programming the UP and DN thresholds as part of the thermal interrupt response. Software should build in some hysteresis when updating thresholds to prevent spurious interrupts. For instance, continuing the previous example of thermal zones: 40ºC-60ºC, 60ºC-80ºC, etc. When software responds to temperature > 60ºC and it sets the new thresholds to be UP=80, DN=60, software will immediately get a spurious DN interrupt if temperature dips below 60ºC. This can be prevented by re-programming UP=80, DN=55 instead which adds some hysteresis for the DN interrupt. **Note: no special hardware support is required for this case.**

40.6.2 Monitors for Thermtrip

For each sensor group, SOC_THERM_thermctl provides a "thermtrip" capability. If enabled, this capability initiates an immediate shutdown when the group's temperature crosses a software-configured threshold. SOC_THERM implements the thresholding but relies on PMC and the boot ROM to implement the shutdown.

Thermtrip is the ultimate backstop for excessively high temperatures. It should only happen in truly exceptional circumstances (when normal thermal management fails to control temperatures). A thermtrip shutdown is NOT graceful.

1. It may cause storage media corruption
2. It provides no mechanism for alerting the user prior to the shutdown

The probability of corrupting storage on a thermtrip may be low, but it is non-zero. The only way to mitigate #1 is to avoid a thermtrip shutdown in the first place. Similarly, the mitigation for #2 is to avoid a thermtrip shutdown. Software can and should configure SOC_THERM (using one of the generic levels) to raise an interrupt at a temperature below the thermtrip threshold. When that interrupts occurs, software should initiate a controlled shutdown.

There is no THERMTRIP functionality for cold throttling (i.e., chip does not shutdown if it's "too cold"). This is a purely software-based thermal throttling solution.

40.6.3 Register Specification

SOC_THERM_thermctl registers are located with all other SOC_THERM registers. Threshold configuration/status registers are defined for each level and each set of temperature sensors. This is denoted below by the N and S in the name, where N=0~3 and S=CPU/GPU/M/TSENSE. Level activation for digital monitor groups may be preconditioned on TSENSE group being active (TS configuration bit).

40.7 Throttling Controller (SOC_THERM_throttlectl)

40.7.1 Introduction

Clock slowdown is the primary method for thermal management. This block serves the following functions:

- Arbitrating between different throttle indicators from other blocks to come up with the correct throttling level.
- Interfacing with the CAR block to perform clock slowdown in a di/dt safe manner.
- Communicating relevant throttling information to software so that clock frequency changes are coordinated.

Throttling is implemented by a combination of SOC_THERM_throttlectl and CAR. A summary of throttling controller function is as follows:

1. SOC_THERM_throttlectl receives all the thermal, EDP and overcurrent alert events from the other detection blocks.
2. Each indicator is associated with a “throttle vector” which contains programmable set of throttle settings for each indicator (thermal, EDP, ac_mode) and for each target (CPU, GPU). In case of multiple indicators firing, arbitration is performed to select final throttle settings.
3. To ensure that the relevant clocks are throttled in a di/dt safe manner, throttling is done gradually. This is done by a “Throttle Sequencer” which sequences through gradually increasing throttle settings to reach the target frequency specified by the throttle vector. The throttle sequencer provides the relevant information to CAR which actually effects the throttling.
4. To guard against di/dt issues, several throttling mechanisms are available. Only pulse skipping (skip M out of every N clock pulses) should be used. Controls for Dynamic Ramping PLL (PLLx) are present in the register space, but should not be used.

40.7.2 Specifying Throttle Settings: Throttle Vectors

Every throttling indicator and throttling target (CPU, GPU) is associated with what is called a “throttle vector”. There are two styles of “throttle vector”, CPU style and GPU style. The CPU style vector is a combination of PLLx and pulse skipper settings to be used when the associated throttle indicator is asserted. The GPU style vector is a 9-bit field (only lowest 3 bits are used, the other 6 bits reserved) to be used to drive the interface to the GPU when the associated throttling indicator is asserted. The throttle vector is configured by software.

As multiple throttling indicators can be active at any time, an arbiter is needed to identify the final throttle settings to be applied. CPU and GPU have separate arbiters. The CPU arbiter will output the PLL and the pulse skippers setting to CPU sequencer and the CPU sequencer will talk to the CAR to throttle the CPU accordingly. The GPU arbiter will directly drive the thermal event interface to the GK20A.

The CPU throttling vector contains settings to achieve a target frequency for a particular throttling indicator. It does not specify the actual clock slowdown factors. These are generated by the CPU throttle sequencer which is described in the next subsection.

There is a throttling vector per target M (= [CPU]) and indicator N, where N can be one of:

- Lite Thermal throttle
- Heavy throttle
- OC1 – OC5
- CPU BA EDP (do not use)
- CPU BA DIDT (do not use)

In addition, there is an additional register for software to configure the pulse skipper settings.

The registers are organized as top-level (used for fields that are common for all throttle vectors), GPU throttle vector register, CPU pulse skipper registers and CPU PLL registers.

GPU throttling is implemented in the GPU’s NV_THERM. SOC_THERM will just drive the thermal event interface to the GPU to trigger the throttling and no longer need to talk to CAR for GPU throttling, so most GPU Pskipper/PLL fields are obsolete.

40.7.2.1 Throttling Vector Arbitration

Since each indicator has its own throttle settings and multiple indicators can be asserted at the same time, an arbiter is needed to select the final throttle settings to be applied to the clocks. It is the software's responsibility to set priorities for the various indicators. Typically, vectors with lower target frequency (i.e., more aggressive throttling) would have higher priority.

SOC_THERM_throttlectl needs a simple max function to pick the throttle vector with the highest priority. Note that in addition to setting the pulse skipper throttle, the final throttle vector also sets other behavior such as HW_RESTORE_EN and LONG_LATENCY_THROTTLE bits.

Software requirements when setting priorities

- Software needs to ensure that critical thermal/OC alarms are given higher priority.
- Software needs to ensure that there is no race if priority of two vectors is set to the same value; i.e., if priority is the same, then the throttle settings should be the same, otherwise it cannot be guaranteed which vector is picked.

Note that the priority is per-indicator-only (not per target). When the arbiter selects a particular priority, it automatically selects everything about the throttle vector:

- All configuration/control registers which define how PLL and pulse skipper are throttled
- All behavior fields such as HW_RESTORE_EN and LONG_LATENCY_THROTTLE
- Priority
- Throttle hysteresis

Even though priority is per-indicator, there is a separate arbiter per throttling target (GPU or CPU). If a throttling indicator does not enable a throttling target (ENB=0), the indicator does not participate in the arbitration for that target.

40.7.3 Pulse Skipper Throttling

40.7.3.1 Software Programming of Pulse Skipper

There are two paths for software to control pulse skippers:

1. **Using the throttle sequencer:** An important difference is that software is now able to control the skippers through the throttle sequencer so that skippers can be correctly sequenced to avoid di/dt issues. Software can program the target M/N values directly to SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_CTRL and SOC_THERM will sequence the skippers to these targets without needing software control. The skipper ramp rate is controlled by fields in SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_RAMP_RATE.
2. **Direct control (similar to control through CAR):** Software still has the option to directly set pulse skipper config by setting the SEQ_BYPASS_MODE bit in SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_RAMP_RATE[31]) which is used to bypass the throttle sequencer.

The software pulse skipper configuration is also used as the reference to restore the skippers to their un-throttled state.

40.7.3.2 Pulse Skipper Restore Mechanism

Since pulse skippers are primarily meant to be controlled by hardware, SOC_THERM provides a mechanism to automatically restore these to their un-throttled state when there are no active throttle indicators. Note that restoring also needs to happen in a di/dt safe manner to avoid an instantaneous change in frequency.

Pulse skippers are restored regardless of the HW_RESTORE_EN behavior of the final throttle vector if PSKIP_RESTORE_CTL=='0'. In this case, SOC_THERM ramps the skippers to the config programmed in SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_CTRL.

40.7.3.3 Changing N-Value (Divisor)

SOC_THERM pulse skipper divisor value should be fixed at 256 (default).

40.7.4 Recovering from Throttled State

Once a throttling event has passed, the system needs to recover from the throttled state. The recovery mechanism is based on the type of throttle event.

40.7.4.1 Recovery from Thermal Throttling Events

In case of thermal events, software will be responsible for recovering from the throttled state. Both, Lite and Heavy, thermal throttling events result in an interrupt which software can use to initiate recovery. In the case of heavy hardware throttling, software cannot explicitly clear the interrupt – it is cleared when the temperature falls below the heavy hardware throttling threshold. Software can disable Lite hardware throttling. In either case, software has the knowledge of every thermal throttling event and can recover from the throttled state.

During recovery, software may want to know the throttled frequency. It can get this information by looking at one of the SOC_THERM_throttlectl status registers.

It is possible that another thermal throttling event occurs while software is trying to recover the system. In such cases, it is possible that the system ping-pongs between throttled and non-throttled states. This can be addressed somewhat by allowing throttling to continue for a longer duration to make sure that the temperature is lowered sufficiently so that software can recover the system successfully.

40.7.4.2 Recovery from EDP and Over-current Events

Recovery from Brief Throttling Mode

In brief throttling mode, hardware detects an EDP or over-current event and throttles the system for a certain time interval. In this mode, hardware is responsible for restoring the system to its pre-throttled state. This is done by rolling back the steps that the hardware took to throttle the system:

Recovery from Sticky Throttling Mode

In sticky throttling mode, hardware continues to throttle the system until software chooses to disable throttling. In this case, it is software's responsibility to recover the system to a non-throttled state.

40.8 Programming Guidance

40.8.1 Thermal Management

Software has a broad role in thermal management, both setting up and managing the thermal sensing/throttling hardware and implementing software-only thermal management policies. This section provides guidance only on setting up and using SOC_THERM hardware features.

40.8.2 Requirements Summary

To utilize SOC_THERM for thermal management, software must implement the following support functions:

1. Initialize the SOC_THERM Controller
    a. Program clocks (soc_therm_clk and tsensor_clk) and do basic controller initialization
    b. Initialize thermal sensors based on per-chip calibration values located in fuses. This is essential to get accurate temperature readings.
    c. Set appropriate thermal shutdown threshold and enable thermal shutdown function in SOC_THERM and PMC.
    d. Set appropriate temperature thresholds for interrupts to software. These thresholds are based on thermal policy, but must be margined as appropriate for sensor error and hotspot offset.
    e. Set appropriate temperature thresholds and throttling amounts for hardware thermal throttling. Hardware thermal throttling is a useful, but optional, addition to high-temperature thermal management policy. If used it must be configured.

2. Respond to SOC_THERM temperature interrupts as appropriate, potentially reprogramming interrupt thresholds to implement thermal management policies.
3. Manage limitations related to TSOSC Vmin and/or CPU/GPU rails being powered-off, using fallback mechanisms to use PLLX_TSOSC when necessary.

40.8.3 Clocks Configuration

Tegra X1 clock and reset unit (CAR) provides a clock source mux and a clock divider for configuring tsensor_clk. See the CLK_RST_CONTROLLER_CLK_SOURCE_TSENSOR_0 register in the Clock and Reset Controller chapter. The CAR allows tsensor_clock to be sourced from PLLP, PLLC, CLK_M, or CLK_S. Software should use CLK_M, which runs at 38.4 MHz in most cases, but 12 MHz is also an option. The tsensor_clk frequency must run at 400 kHz. The frequency can be reached from either a 12MHz XTAL (CLK_M_DIVISOR=1, TSENSOR_CLK_DIVISOR=30) or a 38.4 MHz XTAL (one of CLK_M_DIVISOR={1,2,3}, TSENSOR_CLK_DIVISOR={96,48,32}).

40.8.4 Thermal Sensor Initialization

This is covered in earlier sections, but to summarize:

For each TSOSC:

1. Read and decode fuses
2. Calculate and program ThermA/B coefficients

For each sensing group

1. Program offsets for fallback to PLLX_TSOSC
2. Program sensor thresholds, including margins for accuracy and hotspot offset
3. Arm thermal shutdown as appropriate
4. Enable thermal sensing

40.9 TSensor Registers

Refer to Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

All SOC_THERM registers are in the APB address space. Refer to Chapter 2: Address Map for the base addresses.

40.9.1 SOC_THERM_THERMCTL_LEVEL0_GROUP_CPU_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY

Bit	R/W	Reset	Description
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.2 SOC_THERM_THERMCTL_LEVEL0_GROUP_GPU_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.3 SOC_THERM_THERMCTL_LEVEL0_GROUP_MEM_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.4 SOC_THERM_THERMCTL_LEVEL0_GROUP_TSENSE_0

Offset: 0xc | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000x00000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.5 SOC_THERM_THERMCTL_LEVEL0_UP_STATS_0

Offset: 0x10 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

40.9.6 SOC_THERM_THERMCTL_LEVEL0_DN_STATS_0

Offset: 0x14 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

40.9.7 SOC_THERM_THERMCTL_LEVEL1_GROUP_CPU_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY

Bit	R/W	Reset	Description
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.8 SOC_THERM_THERMCTL_LEVEL1_GROUP_GPU_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.9 SOC_THERM_THERMCTL_LEVEL1_GROUP_MEM_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS:TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.10 SOC_THERM_THERMCTL_LEVEL1_GROUP_TSENSE_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000x00000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.11 SOC_THERM_THERMCTL_LEVEL1_UP_STATS_0

Offset: 0x30 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: count for UP threshold breaches for this level used in the lab

40.9.12 SOC_THERM_THERMCTL_LEVEL1_DN_STATS_0

Offset: 0x34 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: count for DN threshold breaches for this level used in the lab

40.9.13 SOC_THERM_THERMCTL_LEVEL2_GROUP_CPU_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY

Bit	R/W	Reset	Description
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.14 SOC_THERM_THERMCTL_LEVEL2_GROUP_GPU_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.15 SOC_THERM_THERMCTL_LEVEL2_GROUP_MEM_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.16 SOC_THERM_THERMCTL_LEVEL2_GROUP_TSENSE_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000x00000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.17 SOC_THERM_THERMCTL_LEVEL2_UP_STATS_0

Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

40.9.18 SOC_THERM_THERMCTL_LEVEL2_DN_STATS_0

Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

40.9.19 SOC_THERM_THERMCTL_LEVEL3_GROUP_CPU_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY

Bit	R/W	Reset	Description
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.20 SOC_THERM_THERMCTL_LEVEL3_GROUP_GPU_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.21 SOC_THERM_THERMCTL_LEVEL3_GROUP_MEM_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS:TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.22 SOC_THERM_THERMCTL_LEVEL3_GROUP_TSENSE_0

Offset: 0x6c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxx00000000000000000000x00000xx)

Bit	R/W	Reset	Description
26:18	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
17:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC in 0.5ºC increments
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.9.23 SOC_THERM_THERMCTL_LEVEL3_UP_STATS_0

Offset: 0x70 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

40.9.24 SOC_THERM_THERMCTL_LEVEL3_DN_STATS_0

Offset: 0x74 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

40.9.25 SOC_THERM_THERMCTL_THERMTRIP_CTL_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x8349a4d2 (0b10000011010010011010010011010010)

Bit	Reset	Description
31	0x1	ANY_EN: Initiate THERMTRIP based on any monitoring group 0 = OFF 1 = ON
30	0x0	MEM_EN: Initiate THERMTRIP based on MEM monitoring group 0 = OFF 1 = ON
29	0x0	GPU_EN: Initiate THERMTRIP based on GPU monitoring group 0 = OFF 1 = ON
28	0x0	CPU_EN: Initiate THERMTRIP based on CPU monitoring group 0 = OFF 1 = ON
27	0x0	TSENSE_EN: Initiate THERMTRIP based on TSENSE monitoring group
26:18	0xd2	GPU_N_MEM: Threshold for thermal shutdown for GPU group - 1/2 degree precision
17:9	0xd2	CPU: Threshold for thermal shutdown for CPU group - 1/2 degree precision
8:0	0xd2	TSENSE: Threshold for thermal shutdown for TSENSE group - 1/2 degree precision

40.9.26 SOC_THERM_THERMCTL_INTR_STATUS_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt for level 1 0 = OFF 1 = ON
26	0x0	MU1:MEM group up threshold interrupt for level 1 0 = OFF 1 = ON
25	0x0	MD0:MEM group down threshold interrupt for level 0 0 = OFF 1 = ON
24	0x0	MU0:MEM group up threshold interrupt for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt for level 3 0 = OFF 1 = ON
22	0x0	GU3:GPU group up threshold interrupt for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt for level 2 0 = OFF 1 = ON
20	0x0	GU2:GPU group up threshold interrupt for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt for level 1 0 = OFF 1 = ON
18	0x0	GU1:GPU group up threshold interrupt for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt for level 0 0 = OFF 1 = ON
16	0x0	GU0:GPU group up threshold interrupt for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt for level 2 0 = OFF 1 = ON

Bit	Reset	Description
12	0x0	CU2:CPU group up threshold interrupt for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt for level 0 0 = OFF 1 = ON
8	0x0	CU0:CPU group up threshold interrupt for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt for level 3 0 = OFF 1 = ON
6	0x0	TU3:TSENSE group up threshold interrupt for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt for level 2 0 = OFF 1 = ON
4	0x0	TU2:TSENSE group up threshold interrupt for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt for level 1 0 = OFF 1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt for level 0 0 = OFF 1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt for level 0 0 = OFF 1 = ON

40.9.27 SOC_THERM_THERMCTL_INTR_EN_0

Offset: 0x88 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt enable for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt enable for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt enable for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt enable for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt enable for level 1 0 = OFF 1 = ON

Bit	Reset	Description
26	0x0	MU1:MEM group up threshold interrupt enable for level 1 0 = OFF 1 = ON
25	0x0	MD0:MEM group down threshold interrupt enable for level 0 0 = OFF 1 = ON
24	0x0	MU0:MEM group up threshold interrupt enable for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt enable for level 3 0 = OFF 1 = ON
22	0x0	GU3:GPU group up threshold interrupt enable for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt enable for level 2 0 = OFF 1 = ON
20	0x0	GU2:GPU group up threshold interrupt enable for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt enable for level 1 0 = OFF 1 = ON
18	0x0	GU1:GPU group up threshold interrupt enable for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt enable for level 0 0 = OFF 1 = ON
16	0x0	GU0:GPU group up threshold interrupt enable for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt enable for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt enable for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt enable for level 2 0 = OFF 1 = ON
12	0x0	CU2:CPU group up threshold interrupt enable for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt enable for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt enable for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt enable for level 0 0 = OFF 1 = ON
8	0x0	CU0:CPU group up threshold interrupt enable for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt enable for level 3 0 = OFF 1 = ON

Bit	Reset	Description
6	0x0	TU3:TSENSE group up threshold interrupt enable for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt enable for level 2 0 = OFF 1 = ON
4	0x0	TU2:TSENSE group up threshold interrupt enable for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt enable for level 1 0 = OFF 1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt enable for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt enable for level 0 0 = OFF 1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt enable for level 0 0 = OFF 1 = ON

40.9.28 SOC_THERM_THERMCTL_INTR_DIS_0

Offset: 0x8c | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31	0x1	MD3:MEM group down threshold interrupt disable for level 3 0 = OFF 1 = ON
30	0x1	MU3:MEM group up threshold interrupt disable for level 3 0 = OFF 1 = ON
29	0x1	MD2:MEM group down threshold interrupt disable for level 2 0 = OFF 1 = ON
28	0x1	MU2:MEM group up threshold interrupt disable for level 2 0 = OFF 1 = ON
27	0x1	MD1:MEM group down threshold interrupt disable for level 1 0 = OFF 1 = ON
26	0x1	MU1:MEM group up threshold interrupt disable for level 1 0 = OFF 1 = ON
25	0x1	MD0:MEM group down threshold interrupt disable for level 0 0 = OFF 1 = ON
24	0x1	MU0:MEM group up threshold interrupt disable for level 0 0 = OFF 1 = ON
23	0x1	GD3:GPU group down threshold interrupt disable for level 3 0 = OFF 1 = ON
22	0x1	GU3:GPU group up threshold interrupt disable for level 3 0 = OFF 1 = ON
21	0x1	GD2:GPU group down threshold interrupt disable for level 2 0 = OFF 1 = ON

Bit	Reset	Description
20	0x1	GU2:GPU group up threshold interrupt disable for level 2 0 = OFF 1 = ON
19	0x1	GD1:GPU group down threshold interrupt disable for level 1 0 = OFF 1 = ON
18	0x1	GU1:GPU group up threshold interrupt disable for level 1 0 = OFF 1 = ON
17	0x1	GD0:GPU group down threshold interrupt disable for level 0 0 = OFF 1 = ON
16	0x1	GU0:GPU group up threshold interrupt disable for level 0 0 = OFF 1 = ON
15	0x1	CD3:CPU group down threshold interrupt disable for level 3 0 = OFF 1 = ON
14	0x1	CU3:CPU group up threshold interrupt disable for level 3 0 = OFF 1 = ON
13	0x1	CD2:CPU group down threshold interrupt disable for level 2 0 = OFF 1 = ON
12	0x1	CU2:CPU group up threshold interrupt disable for level 2 0 = OFF 1 = ON
11	0x1	CD1:CPU group down threshold interrupt disable for level 1 0 = OFF 1 = ON
10	0x1	CU1:CPU group up threshold interrupt disable for level 1 0 = OFF 1 = ON
9	0x1	CD0:CPU group down threshold interrupt disable for level 0 0 = OFF 1 = ON
8	0x1	CU0:CPU group up threshold interrupt disable for level 0 0 = OFF 1 = ON
7	0x1	TD3:TSENSE group down threshold interrupt disable for level 3 0 = OFF 1 = ON
6	0x1	TU3:TSENSE group up threshold interrupt disable for level 3 0 = OFF 1 = ON
5	0x1	TD2:TSENSE group down threshold interrupt disable for level 2 0 = OFF 1 = ON
4	0x1	TU2:TSENSE group up threshold interrupt disable for level 2 0 = OFF 1 = ON
3	0x1	TD1:TSENSE group down threshold interrupt disable for level 1 0 = OFF 1 = ON
2	0x1	TU1:TSENSE group up threshold interrupt disable for level 1 0 = OFF 1 = ON
1	0x1	TD0:TSENSE group down threshold interrupt disable for level 0 0 = OFF 1 = ON

Bit	Reset	Description
0	0x1	TU0:TSENSE group up threshold interrupt disable for level 0 0 = OFF 1 = ON

40.9.29 SOC_THERM_THERMCTL_STATS_CTL_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	CLEAR_DN: Clear DN statistics for all levels
2	0x0	ENB_DN: Enable DN statistics collection for all levels
1	0x0	CLEAR_UP: Clear statistics for all levels
0	0x0	ENB_UP: Enable statistics collection for all levels

40.9.30 SOC_THERM_THERMCTL_SLOWDOWN_THRESHOLD_0

Offset: 0x98 | Read/Write: R/W | Reset: 0x69696969 (0b01101001011010010110100101101001)

Bit	Reset	Description
31:24	0x69	MEM: Slowdown threshold for MEM group
23:16	0x69	GPU: Slowdown threshold for GPU group
15:8	0x69	CPU: Slowdown threshold for CPU group
7:0	0x69	TSENSE: Slowdown threshold for TSENSE group

40.9.31 SOC_THERM_THERMCTL_SLOWDOWN_CTL_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
31:30	0x0	SLOWDOWN_SELECT: Selects which one to throttle 0 = NONE 1 = CPU_ONLY 2 = GPU_ONLY 3 = BOTH
4	0x0	ANY_EN
3	0x0	MEM_EN
2	0x0	GPU_EN
1	0x0	CPU_EN
0	0x0	TSENSE_EN

40.9.32 SOC_THERM_TSENSOR_CPU0_CONFIG0_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: capture overflow 0 = OFF 1 = ON

Bit	Reset	Description
1	0x0	RO_SEL: Reserved
0	0x1	STOP: sensor stop 0 = OFF 1 = ON

40.9.33 SOC_THERM_TSENSOR_CPU0_CONFIG1_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.34 SOC_THERM_TSENSOR_CPU0_CONFIG2_0

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.35 SOC_THERM_TSENSOR_CPU0_STATUS0_0

Offset: 0xcc | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: valid capture available
15:0	X	CAPTURE: last valid raw capture

40.9.36 SOC_THERM_TSENSOR_CPU0_STATUS1_0

Offset: 0xd0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: valid capture available
15:0	X	TEMP: last valid translated temperature

40.9.37 SOC_THERM_TSENSOR_CPU0_STATUS2_0

Offset: 0xd4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.38 SOC_THERM_TSENSOR_CPU1_CONFIG0_0

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics

Bit	Reset	Description
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Reserved
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.9.39 SOC_THERM_TSENSOR_CPU1_CONFIG1_0

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.40 SOC_THERM_TSENSOR_CPU1_CONFIG2_0

Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.41 SOC_THERM_TSENSOR_CPU1_STATUS0_0

Offset: 0xec | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.9.42 SOC_THERM_TSENSOR_CPU1_STATUS1_0

Offset: 0xf0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.9.43 SOC_THERM_TSENSOR_CPU1_STATUS2_0

Offset: 0xf4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.44 SOC_THERM_TSENSOR_CPU2_CONFIG0_0

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Reserved
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.9.45 SOC_THERM_TSENSOR_CPU2_CONFIG1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.46 SOC_THERM_TSENSOR_CPU2_CONFIG2_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.47 SOC_THERM_TSENSOR_CPU2_STATUS0_0

Offset: 0x10c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: valid capture available
15:0	X	CAPTURE: last valid raw capture

40.9.48 SOC_THERM_TSENSOR_CPU2_STATUS1_0

Offset: 0x110 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.9.49 SOC_THERM_TSENSOR_CPU2_STATUS2_0

Offset: 0x114 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.50 SOC_THERM_TSENSOR_CPU3_CONFIG0_0

Offset: 0x120 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Reserved
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.9.51 SOC_THERM_TSENSOR_CPU3_CONFIG1_0

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.52 SOC_THERM_TSENSOR_CPU3_CONFIG2_0

Offset: 0x128 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.53 SOC_THERM_TSENSOR_CPU3_STATUS0_0

Offset: 0x12c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.9.54 SOC_THERM_TSENSOR_CPU3_STATUS1_0

Offset: 0x130 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.9.55 SOC_THERM_TSENSOR_CPU3_STATUS2_0

Offset: 0x134 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.56 SOC_THERM_TSENSOR_MEM0_CONFIG0_0

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Reserved
0	0x1	STOP: sensor stop 0 = OFF 1 = ON

40.9.57 SOC_THERM_TSENSOR_MEM0_CONFIG1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: time between en and capture
20:15	0x0	TIDDQ_EN: time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.58 SOC_THERM_TSENSOR_MEM0_CONFIG2_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.59 SOC_THERM_TSENSOR_MEM0_STATUS0_0

Offset: 0x14c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.9.60 SOC_THERM_TSENSOR_MEM0_STATUS1_0

Offset: 0x150 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.9.61 SOC_THERM_TSENSOR_MEM0_STATUS2_0

Offset: 0x154 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.62 SOC_THERM_TSENSOR_MEM1_CONFIG0_0

Offset: 0x160 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Reserved
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.9.63 SOC_THERM_TSENSOR_MEM1_CONFIG1_0

Offset: 0x164 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.64 SOC_THERM_TSENSOR_MEM1_CONFIG2_0

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.65 SOC_THERM_TSENSOR_MEM1_STATUS0_0

Offset: 0x16c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.9.66 SOC_THERM_TSENSOR_MEM1_STATUS1_0

Offset: 0x170 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.9.67 SOC_THERM_TSENSOR_MEM1_STATUS2_0

Offset: 0x174 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.68 SOC_THERM_TSENSOR_GPU_CONFIG0_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Reserved
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.9.69 SOC_THERM_TSENSOR_GPU_CONFIG1_0

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.70 SOC_THERM_TSENSOR_GPU_CONFIG2_0

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.71 SOC_THERM_TSENSOR_GPU_STATUS0_0

Offset: 0x18c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.9.72 SOC_THERM_TSENSOR_GPU_STATUS1_0

Offset: 0x190 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.9.73 SOC_THERM_TSENSOR_GPU_STATUS2_0

Offset: 0x194 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.74 SOC_THERM_TSENSOR_PLLX_CONFIG0_0

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON

Bit	Reset	Description
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Reserved
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.9.75 SOC_THERM_TSENSOR_PLLX_CONFIG1_0

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: time between en and capture
20:15	0x0	TIDDQ_EN: time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.9.76 SOC_THERM_TSENSOR_PLLX_CONFIG2_0

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.9.77 SOC_THERM_TSENSOR_PLLX_STATUS0_0

Offset: 0x1ac | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.9.78 SOC_THERM_TSENSOR_PLLX_STATUS1_0

Offset: 0x1b0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.9.79 SOC_THERM_TSENSOR_PLLX_STATUS2_0

Offset: 0x1b4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.9.80 SOC_THERM_TSENSOR_PDIV_0

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:12	0x0	CPU_PDIV: PDIV for TS_CPU0 TS_CPU3
11:8	0x0	GPU_PDIV: PDIV for TS_GPU
7:4	0x0	MEM_PDIV: PDIV for TS_MEM
3:0	0x0	PLLX_PDIV: PDIV for TS_PLLX

40.9.81 SOC_THERM_TSENSOR_HOTSPOT_OFF_0

Offset: 0x1c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	CPU_HOTSPOT_OFF:CPU hotspot offset from PLLX
15:8	0x0	GPU_HOTSPOT_OFF:GPU hotspot offset from PLLX
7:0	0x0	MEM_HOTSPOT_OFF:MEM hotspot offset from PLLX

40.9.82 SOC_THERM_TSENSOR_TEMP1_0

Offset: 0x1c8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000))

Bit	Reset	Description
31:16	0x0	CPU_TEMP: Processed CPU temperature seen by thermal throttling logic. Temperature readback format
15:0	0x0	GPU_TEMP: Processed GPU temperature seen by thermal throttling logic. Temperature readback format

Temperature read-back format. Translation will be programmed to produce value in ½ °C units. To make reading easier in read-back registers it is reformatted as:

- Bits 15:8 – temperature absolute value high bits (with ½ °C programming – it will be the temperature in °Celsius
- Bit 7 – LSB. ½ °C bit.
- Bit 0 -- sign bit. 0 indicates positive. 1 indicates negative.

40.9.83 SOC_THERM_TSENSOR_TEMP2_0

Offset: 0x1cc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000))

Bit	Reset	Description
31:16	0x0	MEM_TEMP: Processed MEM temperature seen by thermal throttling logic. Temperature readback format
15:0	0x0	SENSOR_TEMP: Processed sensor (PLLX) temperature seen by thermal throttling logic. Temperature readback format

Temperature read-back format. Translation will be programmed to produce value in ½ °C units. To make reading easier in read-back registers it is reformatted as:

- Bits 15:8 – temperature absolute value high bits (with ½ °C programming – it will be the temperature in °Celsius
- Bit 7 – LSB. ½ °C bit.
- Bit 0 -- sign bit. 0 indicates positive. 1 indicates negative.

40.9.84 SOC_THERM_TSENSOR_TSENSOR_PWR_VLD_OVERRIDE_0

Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	INVALIDATE_ON_PWR_GATING: Use CPU/GPU virtual power gating status to invalidate the CPU/GPU sensors if set otherwise use Rail gating status

40.9.85 SOC_THERM_TSENSOR_SPARE_ECO_0

Offset: 0x1d4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	Reserved.

40.9.86 SOC_THERM_TSENSOR_TEMP_SW_OVERRIDE_0

Offset: 0x1d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SW_OVERRIDE_EN: 1: Enable software override of TSENSOR TEMP registers.

40.9.87 SOC_THERM_TSENSOR_TSENSOR_CLKEN_0

Offset: 0x1dc | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7	0x1	CPU0_CLKEN: 1: Enable soc therm/tsensor clock for CPU0 tsensor logic
6	0x1	CPU1_CLKEN: 1: Enable soc therm/tsensor clock for CPU1 tsensor logic
5	0x1	CPU2_CLKEN: 1: Enable soc therm/tsensor clock for CPU2 tsensor logic
4	0x1	CPU3_CLKEN: 1: Enable soc therm/tsensor clock for CPU3 tsensor logic
3	0x1	GPU_CLKEN: 1: Enable soc therm/tsensor clock for GPU tsensor logic
2	0x1	MEM0_CLKEN: 1: Enable soc therm/tsensor clock for MEM0 tsensor logic
1	0x1	MEM1_CLKEN: 1: Enable soc therm/tsensor clock for MEM1 tsensor logic
0	0x1	PLLX_CLKEN: 1: Enable soc therm/tsensor clock for PLLX tsensor logic

40.9.88 SOC_THERM_TSENSOR_VALID_0

Offset: 0x1e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
11	0x0	MEM1_INVALID: invalidate MEM1 TSOSC
10	0x0	MEM0_INVALID: invalidate MEM0 TSOSC
9	0x0	GPU_INVALID: invalidate GPU TSOSC
8	0x0	PLLX_INVALID: invalidate PLLX TSOSC
3	0x0	CPU3_INVALID: invalidate CPU3 TSOSC
2	0x0	CPU2_INVALID: invalidate CPU2 TSOSC
1	0x0	CPU1_INVALID: invalidate CPU1 TSOSC
0	0x0	CPU0_INVALID: invalidate CPU0 TSOSC

40.9.89 SOC_THERM_TSENSOR_HW_PLLX_OFFSETTING_0

Offset: 0x1e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	EN_HW_PLLX_OFFSET_MEM: enable hardware calculated PLLX offset for MEM TSOSC
1	0x0	EN_HW_PLLX_OFFSET_GPU: enable hardware calculated PLLX offset for GPU TSOSC
0	0x0	EN_HW_PLLX_OFFSET_CPU: enable hardware calculated PLLX offset for CPU TSOSC

40.9.90 SOC_THERM_TSENSOR_PLLX_OFFSET_MIN_0

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	MIN_MEM_OFFSET: Minimum offset allowed for hardware calculated PLLX offset
15:8	0x0	MIN_GPU_OFFSET: Minimum offset allowed for hardware calculated PLLX offset
7:0	0x0	MIN_CPU_OFFSET: Minimum offset allowed for hardware calculated PLLX offset

40.9.91 SOC_THERM_TSENSOR_PLLX_OFFSET_MAX_0

Offset: 0x1ec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	MAX_MEM_OFFSET: Maximum offset allowed for hardware calculated PLLX offset
15:8	0x0	MAX_GPU_OFFSET: Maximum offset allowed for hardware calculated PLLX offset
7:0	0x0	MAX_CPU_OFFSET: Maximum offset allowed for hardware calculated PLLX offset

40.9.92 SOC_THERM_EDP_OC_ALARM_OC1_CFG_0

Offset: 0x310 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.9.93 SOC_THERM_EDP_OC_ALARM_OC1_CNT_THRESHOLD_0

Offset: 0x314 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.9.94 SOC_THERM_EDP_OC_ALARM_OC1_THROTTLE_PERIOD_0

Offset: 0x318 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds. Valid only for brief throttle

40.9.95 SOC_THERM_EDP_OC_ALARM_OC1_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

Offset: 0x31c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.9.96 SOC_THERM_EDP_OC_ALARM_OC1_FILTER_0

Offset: 0x320 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.9.97 SOC_THERM_EDP_OC_ALARM_OC2_CFG_0

Offset: 0x324 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000100)

Bit	Reset	Description
7	0x0	MODEM_EDP_SEL: Used to define the throttling mode of the OC2 input 0 = OC 1 = EDP
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.9.98 SOC_THERM_EDP_OC_ALARM_OC2_CNT_THRESHOLD_0

Offset: 0x328 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.9.99 SOC_THERM_EDP_OC_ALARM_OC2_THROTTLE_PERIOD_0

Offset: 0x32c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds. Valid only for brief throttle

40.9.100 SOC_THERM_EDP_OC_ALARM_OC2_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

Offset: 0x330 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.9.101 SOC_THERM_EDP_OC_ALARM_OC2_FILTER_0

Offset: 0x334 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.9.102 SOC_THERM_EDP_OC_ALARM_OC3_CFG_0

Offset: 0x338 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000100)

Bit	Reset	Description
7	0x0	MODEM_EDP_SEL: Used to define the throttling mode of OC2 input 0 = OC 1 = EDP
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.9.103 SOC_THERM_EDP_OC_ALARM_OC3_CNT_THRESHOLD_0

Offset: 0x33c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.9.104 SOC_THERM_EDP_OC_ALARM_OC3_THROTTLE_PERIOD_0

Offset: 0x340 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds- valid only for brief throttle

40.9.105 SOC_THERM_EDP_OC_ALARM_OC3_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

Offset: 0x344 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.9.106 SOC_THERM_EDP_OC_ALARM_OC3_FILTER_0

Offset: 0x348 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.9.107 SOC_THERM_EDP_OC_ALARM_OC4_CFG_0

Offset: 0x34c | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000100)

Bit	Reset	Description
7	0x0	MODEM_EDP_SEL: Used to define the throttling mode of OC2 input 0 = OC 1 = EDP
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.9.108 SOC_THERM_EDP_OC_ALARM_OC4_CNT_THRESHOLD_0

Offset: 0x350 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.9.109 SOC_THERM_EDP_OC_ALARM_OC4_THROTTLE_PERIOD_0

Offset: 0x354 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds- valid only for brief throttle

40.9.110 SOC_THERM_EDP_OC_ALARM_OC4_COUNT_0

Hardware uses the COUNT register to create a throttling event (by comparing with threshold). This register is reset every time the FILTER timer expires.

Offset: 0x358 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.9.111 SOC_THERM_EDP_OC_ALARM_OC4_FILTER_0

Offset: 0x35c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.9.112 SOC_THERM_EDP_OC_ALARM_OC5_CFG_0

Offset: 0x360 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

This register is reserved.

40.9.113 SOC_THERM_EDP_OC_ALARM_OC5_CNT_THRESHOLD_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

This register is reserved.

40.9.114 SOC_THERM_EDP_OC_ALARM_OC5_THROTTLE_PERIOD_0

Offset: 0x368 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

This register is reserved.

40.9.115 SOC_THERM_EDP_OC_ALARM_OC5_COUNT_0

Offset: 0x36c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

This register is reserved.

40.9.116 SOC_THERM_EDP_OC_ALARM_OC5_FILTER_0

Offset: 0x370 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

This register is reserved.

40.9.117 SOC_THERM_EDP_OC_INTR_STATUS_0

Offset: 0x39c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT:CLDVFS interrupt status
4	0x0	OC5: Reserved.
3	0x0	OC4: OC event 4 interrupt status

Bit	Reset	Description
2	0x0	OC3: OC event 3 interrupt status
1	0x0	OC2: OC event 2 interrupt status
0	0x0	OC1: OC event 1 interrupt status

40.9.118 SOC_THERM_EDP_OC_INTR_ENABLE_0

Offset: 0x3a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT_EN:CLDVFS interrupt enable
4	0x0	OC5_EN: OC Reserved
3	0x0	OC4_EN: OC event 4 interrupt enable
2	0x0	OC3_EN: OC event 3 interrupt enable
1	0x0	OC2_EN: OC event 2 interrupt enable
0	0x0	OC1_EN: OC event 1 interrupt enable

40.9.119 SOC_THERM_EDP_OC_INTR_DISABLE_0

Offset: 0x3a4 | Read/Write: R/W | Reset: 0x0013001f (0bxxxxxxxxxxx1xx11xxxxxxxxxx11111)

Bit	Reset	Description
20	0x1	CLDVFS_DIDT_DIS:CLDVFS interrupt disable
4	0x1	OC5_DIS: Reserved
3	0x1	OC4_DIS: OC event 4 interrupt disable
2	0x1	OC3_DIS: OC event 3 interrupt disable
1	0x1	OC2_DIS: OC event 2 interrupt disable
0	0x1	OC1_DIS: OC event 1 interrupt disable

40.9.120 SOC_THERM_EDP_OC_ALARM_OC1_STATS_0

This register counts the number of over-current (OC) occurrences until it is cleared by software.

Offset: 0x3a8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.9.121 SOC_THERM_EDP_OC_ALARM_OC2_STATS_0

This register counts the number of over-current (OC) occurrences until it is cleared by software.

Offset: 0x3ac | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.9.122 SOC_THERM_EDP_OC_ALARM_OC3_STATS_0

This register counts the number of over-current (OC) occurrences until it is cleared by software.

Offset: 0x3b0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.9.123 SOC_THERM_EDP_OC_ALARM_OC4_STATS_0

This register counts the number of over-current (OC) occurrences until it is cleared by software.

Offset: 0x3b4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.9.124 SOC_THERM_EDP_OC_ALARM_OC5_STATS_0

Offset: 0x3b8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

This register is reserved.

40.9.125 SOC_THERM_EDP_OC_ALARM_STATS_CTRL_0

Offset: 0x3c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CLEAR_ALL: Clear all statistics
0	0x0	ENB_ALL: Enable all statistics collections for all counters

40.9.126 SOC_THERM_EDP_OC_CLDVFS_DIDT_CNT_THRESHOLD_0

Offset: 0x3c8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.9.127 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_COUNT_0

Offset: 0x3cc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.9.128 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_CNT_FILTER_0

Offset: 0x3d0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: event count

40.9.129 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_STATS_0

Offset: 0x3d4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.9.130 SOC_THERM_EDP_OC_ALARM_THROTTLE_PERIOD_CTL_0

Offset: 0x3d8 | Read/Write: R/W | Reset: 0x000000cc (0bxxxxxxxxxxxxxxxxxxxxxxxx11001100))

Bit	Reset	Description
7:0	0xcc	NUM_CLKS_IN_1US: Number of soc_therm clocks in 1µs used to determine brief throttle length (default based on 204 MHz)

40.9.131 SOC_THERM_EDP_OC_OC2_MODEM_EDP_VALUE_0

Offset: 0x3dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MODEM_EDP_VALUE: EDP count corresponding to the incremental current draw by modem subsystem

40.9.132 SOC_THERM_EDP_OC_OC3_MODEM_EDP_VALUE_0

Offset: 0x3e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MODEM_EDP_VALUE: EDP count corresponding to the incremental current draw by modem subsystem

40.9.133 SOC_THERM_EDP_OC_OC4_MODEM_EDP_VALUE_0

Offset: 0x3e4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MODEM_EDP_VALUE: EDP count corresponding to the incremental current draw by modem subsystem

40.9.134 SOC_THERM_THROTTLECTL_GLOBAL_THROTTLE_CFG_0

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000011 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx10001)

Bit	Reset	Description
4	0x1	LEGACY_PSKIP_MODE:1: (default) pulse skipper will not transition to lower throttling when higher throttling alarm deasserts while a
3	0x0	DFLL_PSKIP_CTRL:0: (default) pause CPU pulse skipper when cldvfs2soc_therm_skipper1_en is asserted1: do not pause CPU pulse skipper
2	0x0	PSKIP_RESTORE_CTRL:0: Software does not restore (default) 1: Software restores. If HW_RESTORE_EN==0, hardware will not restore the pulse skipper
1	0x0	SW_OVERRIDE_MODE: Pulse skipper software override:=1 causes all throttle indicators to use software pulse skipper cfg for throttling pulse skipper 0 = DISABLE 1 = ENABLE
0	0x1	ENB: Single bit to disable all throttling 0 = DISABLE 1 = ENABLE

40.9.135 SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_CTRL_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.136 SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_RAMP_RATE_0

Offset: 0x40c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles

Bit	Reset	Description
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.137 SOC_THERM_THROTTLECTL_SW_GPU_PSKIP_CTRL_0

Offset: 0x410 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000)

Bit	Reset	R/W	Description
31	0x0	RW	Reserved
30	0x1	RO	LOW_MED_HIGH: This field is set to 1'b1 by hardware after reset. Software needs to read this field to decide the style of GPU throttling. When this bit is read back as 1'b1, this means the LOW/MEDIUM/HIGH interface is used. When this bit is read back as 1'b0, it means the legacy GPU throttling with PSKIP is used.
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding).
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.138 SOC_THERM_THROTTLECTL_SW_GPU_PSKIP_RAMP_RATE_0

Offset: 0x414 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLED 1 = ENABLED
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.139 SOC_THERM_THROTTLECTL_CPU_PSKIP_STATUS_0

Offset: 0x418 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:12	X	CURR_PULSE_SKIP_M:
11:4	X	CURR_PULSE_SKIP_N:
1	X	SW_OVERRIDE_STATUS: 0 = DISABLED 1 = ENABLED
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

40.9.140 SOC_THERM_THROTTLECTL_GPU_PSKIP_STATUS_0

Offset: 0x41c | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
22:20	X	THROTTLE_DEPTH
19:12	X	CURR_PULSE_SKIP_M:
11:4	X	CURR_PULSE_SKIP_N
1	X	SW_OVERRIDE_STATUS: 0 = DISABLED 1 = ENABLED
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

40.9.141 SOC_THERM_THROTTLECTL_PRIORITY_LOCK_0

Offset: 0x424 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Maximum priority allowed for software programmable vectors

40.9.142 SOC_THERM_THROTTLECTL_THROTTLE_STATUS_0

Offset: 0x428 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12	X	PRIORITY_LOCK_BREACH: Set to '1' if the selected throttle vector is of higher priority than the limit specified by the Boot Loader. The field is cleared when software updates PRIORITY registers.
11:4	X	THROTTLE_SEQ_STATE:11:8 = gpu sqr, 7:4 = cpu sqr 0 = WAIT_SQR 1 = START_ARB 2 = PLLX_THROTTLE 3 = DFLL_THROTTLE 4 = PSKIP_THROTTLE 5 = THROTTLE_DONE 6 = PSKIP_RESTORE 7 = DFLL_RESTORE 8 = PLLX_RESTORE 9 = RESTORE_DONE
0	X	ENB_STATUS: Global enable status 0 = DISABLED 1 = ENABLED

40.9.143 SOC_THERM_THROTTLECTL_CPU_DFLL_STATUS_0

Offset: 0x42c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:5	X	THROTTLE_INDEX_STATUS:
4	X	HW_RESTORE_EN: 0 = DISABLED 1 = ENABLED
3	X	HW_THROTTLE_EN: 0 = DISABLED 1 = ENABLED
2	X	HW_RAMP_STATUS: 0 = RAMP_DONE 1 = IN_PROGRESS
1	X	SW_OVERRIDE_STATUS: 0 = DISABLED 1 = ENABLED
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

40.9.144 SOC_THERM_THROTTLECTL_LITE_CPU_PSKIP_CTRL_0

Offset: 0x430 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.145 SOC_THERM_THROTTLECTL_LITE_CPU_PSKIP_RAMP_RATE_0

Offset: 0x434 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.146 SOC_THERM_THROTTLECTL_LITE_GPU_PSKIP_CTRL_0

Offset: 0x438 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000)

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.147 SOC_THERM_THROTTLECTL_LITE_GPU_PSKIP_RAMP_RATE_0

Offset: 0x43c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.148 SOC_THERM_THROTTLECTL_LITE_THROTTLE_PRIORITY_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.9.149 SOC_THERM_THROTTLECTL_LITE_THROTTLE_DELAY_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.9.150 SOC_THERM_THROTTLECTL_LITE_CPU_DFLL_CTRL_0

Offset: 0x44c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	DFLL_THROTTLE_INDEX: One hot Indication of how much throttling to perform 3'b000: No throttling (Default) 3'b001: Light throttling 3'b010: Medium throttling 3'b100: Heavy throttling
2	0x0	SW_OVERRIDE_MODE: 1: enable software override mode (SOC THERM does not assert freq_vld) 0 = DISABLE 1 = ENABLE
1	0x0	DFLL_RAMP_MODE: 0: don't wait for ramp_done signal from DFLL 1: wait for ramp_done signal from DFLL 0 = IGNORE_RAMP_DONE 1 = WAIT_FOR_RAMP_DONE
0	0x0	ENB: Enable DFLL throttling for this indicator 0 = DISABLE 1 = ENABLE

40.9.151 SOC_THERM_THROTTLECTL_HEAVY_CPU_PSKIP_CTRL_0

Offset: 0x460 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.152 SOC_THERM_THROTTLECTL_HEAVY_CPU_PSKIP_RAMP_RATE_0

Offset: 0x464 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.153 SOC_THERM_THROTTLECTL_HEAVY_GPU_PSKIP_CTRL_0

Offset: 0x468 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.154 SOC_THERM_THROTTLECTL_HEAVY_GPU_PSKIP_RAMP_RATE_0

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.155 SOC_THERM_THROTTLECTL_HEAVY_THROTTLE_PRIORITY_0

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.9.156 SOC_THERM_THROTTLECTL_HEAVY_THROTTLE_DELAY_0

Offset: 0x478 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.9.157 SOC_THERM_THROTTLECTL_HEAVY_CPU_DFLL_CTRL_0

Offset: 0x47c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	DFLL_THROTTLE_INDEX: One hot Indication of how much throttling to perform 3'b000: No throttling (Default) 3'b001: Light throttling 3'b010: Medium throttling 3'b100: Heavy throttling
2	0x0	SW_OVERRIDE_MODE: 1: enable software override mode (SOC THERM does not assert freq_vld) 0 = DISABLE 1 = ENABLE
1	0x0	DFLL_RAMP_MODE: 0: don't wait for ramp_done signal from DFLL 1: wait for ramp_done signal from DFLL 0 = IGNORE_RAMP_DONE 1 = WAIT_FOR_RAMP_DONE
0	0x0	ENB: Enable DFLL throttling for this indicator 0 = DISABLE 1 = ENABLE

40.9.158 SOC_THERM_THROTTLECTL_OC1_CPU_PSKIP_CTRL_0

Offset: 0x490 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.159 SOC_THERM_THROTTLECTL_OC1_CPU_PSKIP_RAMP_RATE_0

Offset: 0x494 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.160 SOC_THERM_THROTTLECTL_OC1_GPU_PSKIP_CTRL_0

Offset: 0x498 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.161 SOC_THERM_THROTTLECTL_OC1_GPU_PSKIP_RAMP_RATE_0

Offset: 0x49c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.162 SOC_THERM_THROTTLECTL_OC1_THROTTLE_PRIORITY_0

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.9.163 SOC_THERM_THROTTLECTL_OC1_THROTTLE_DELAY_0

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.9.164 SOC_THERM_THROTTLECTL_OC1_CPU_DFLL_CTRL_0

Offset: 0x4ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	DFLL_THROTTLE_INDEX: One hot Indication of how much throttling to perform 3'b000: No throttling (Default) 3'b001: Light throttling 3'b010: Medium throttling 3'b100: Heavy throttling
2	0x0	SW_OVERRIDE_MODE: 1: enable SW override mode (SOC THERM does not assert freq_vld) 0 = DISABLE 1 = ENABLE
1	0x0	DFLL_RAMP_MODE: 0: don't wait for ramp_done signal from DFLL 1: wait for ramp_done signal from DFLL 0 = IGNORE_RAMP_DONE 1 = WAIT_FOR_RAMP_DONE
0	0x0	ENB: Enable DFLL throttling for this indicator 0 = DISABLE 1 = ENABLE

40.9.165 SOC_THERM_THROTTLECTL_OC2_CPU_PSKIP_CTRL_0

Offset: 0x4c0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.166 SOC_THERM_THROTTLECTL_OC2_CPU_PSKIP_RAMP_RATE_0

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.167 SOC_THERM_THROTTLECTL_OC2_GPU_PSKIP_CTRL_0

Offset: 0x4c8 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx0000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.168 SOC_THERM_THROTTLECTL_OC2_GPU_PSKIP_RAMP_RATE_0

Offset: 0x4cc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.169 SOC_THERM_THROTTLECTL_OC2_THROTTLE_PRIORITY_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.9.170 SOC_THERM_THROTTLECTL_OC2_THROTTLE_DELAY_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.9.171 SOC_THERM_THROTTLECTL_OC2_CPU_DFLL_CTRL_0

Offset: 0x4dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	DFLL_THROTTLE_INDEX: One hot Indication of how much throttling to perform 3'b000: No throttling (Default) 3'b001: Light throttling 3'b010: Medium throttling 3'b100: Heavy throttling
2	0x0	SW_OVERRIDE_MODE: 1: enable SW override mode (SOC THERM does not assert freq_vld) 0 = DISABLE 1 = ENABLE
1	0x0	DFLL_RAMP_MODE: 0: don't wait for ramp_done signal from DFLL 1: wait for ramp_done signal from DFLL 0 = IGNORE_RAMP_DONE 1 = WAIT_FOR_RAMP_DONE
0	0x0	ENB: Enable DFLL throttling for this indicator 0 = DISABLE 1 = ENABLE

40.9.172 SOC_THERM_THROTTLECTL_OC3_CPU_PSKIP_CTRL_0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.173 SOC_THERM_THROTTLECTL_OC3_CPU_PSKIP_RAMP_RATE_0

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.174 SOC_THERM_THROTTLECTL_OC3_GPU_PSKIP_CTRL_0

Offset: 0x4f8 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.175 SOC_THERM_THROTTLECTL_OC3_GPU_PSKIP_RAMP_RATE_0

Offset: 0x4fc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.176 SOC_THERM_THROTTLECTL_OC3_THROTTLE_PRIORITY_0

Offset: 0x504 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.9.177 SOC_THERM_THROTTLECTL_OC3_THROTTLE_DELAY_0

Offset: 0x508 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.9.178 SOC_THERM_THROTTLECTL_OC3_CPU_DFLL_CTRL_0

Offset: 0x50c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	DFLL_THROTTLE_INDEX: One hot Indication of how much throttling to perform 3'b000: No throttling (Default) 3'b001: Light throttling 3'b010: Medium throttling 3'b100: Heavy throttling
2	0x0	SW_OVERRIDE_MODE: 1: enable software override mode (SOC THERM does not assert freq_vld) 0 = DISABLE 1 = ENABLE
1	0x0	DFLL_RAMP_MODE: 0: don't wait for ramp_done signal from DFLL 1: wait for ramp_done signal from DFLL 0 = IGNORE_RAMP_DONE 1 = WAIT_FOR_RAMP_DONE
0	0x0	ENB: Enable DFLL throttling for this indicator 0 = DISABLE 1 = ENABLE

40.9.179 SOC_THERM_THROTTLECTL_OC4_CPU_PSKIP_CTRL_0

Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.180 SOC_THERM_THROTTLECTL_OC4_CPU_PSKIP_RAMP_RATE_0

Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.181 SOC_THERM_THROTTLECTL_OC4_GPU_PSKIP_CTRL_0

Offset: 0x528 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx0000000000000000000))

Bit	Reset	R/W	Description
31	0x0	RW	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
30	X	RO	LOW_MED_HIGH: Throttling configured through THROTTLE_DEPTH 0 = DISABLE 1 = ENABLE
18:16	0x0	RW	THROTTLE_DEPTH: Configures LOW/MEDIUM/HIGH throttling (zero-one-hot encoding)
15:8	0x0	RW	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	RW	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.182 SOC_THERM_THROTTLECTL_OC4_GPU_PSKIP_RAMP_RATE_0

Offset: 0x52c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.9.183 SOC_THERM_THROTTLECTL_OC4_THROTTLE_PRIORITY_0

Offset: 0x534 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.9.184 SOC_THERM_THROTTLECTL_OC4_THROTTLE_DELAY_0

Offset: 0x538 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.9.185 SOC_THERM_THROTTLECTL_OC4_CPU_DFLL_CTRL_0

Offset: 0x53c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	DFLL_THROTTLE_INDEX: One hot Indication of how much throttling to perform 3'b000: No throttling (Default) 3'b001: Light throttling 3'b010: Medium throttling 3'b100: Heavy throttling
2	0x0	SW_OVERRIDE_MODE: 1: enable software override mode (SOC THERM does not assert freq_vld) 0 = DISABLE 1 = ENABLE
1	0x0	DFLL_RAMP_MODE: 0: don't wait for ramp_done signal from DFLL 1: wait for ramp_done signal from DFLL 0 = IGNORE_RAMP_DONE 1 = WAIT_FOR_RAMP_DONE
0	0x0	ENB: Enable DFLL throttling for this indicator 0 = DISABLE 1 = ENABLE

40.9.186 SOC_THERM_THROTTLECTL_OC5_CPU_PSKIP_CTRL_0

Offset: 0x550 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

This register is reserved.

40.9.187 SOC_THERM_THROTTLECTL_OC5_CPU_PSKIP_RAMP_RATE_0

Offset: 0x554 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

This register is reserved.

40.9.188 SOC_THERM_THROTTLECTL_OC5_GPU_PSKIP_CTRL_0

Offset: 0x558 | Read/Write: R/W | Reset: 0xX0000000 (0b0xxxxxxxxxxxx00000000000000000000)

This register is reserved.

40.9.189 SOC_THERM_THROTTLECTL_OC5_GPU_PSKIP_RAMP_RATE_0

Offset: 0x55c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

This register is reserved.

40.9.190 SOC_THERM_THROTTLECTL_OC5_THROTTLE_PRIORITY_0

Offset: 0x564 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

This register is reserved.

40.9.191 SOC_THERM_THROTTLECTL_OC5_CPU_DFLL_CTRL_0

Offset: 0x56c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

This register is reserved.

40.9.192 SOC_THERM_THROTTLECTL2_CPU_DFLL_RAMP_TIMER_0

Offset: 0x5e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DFLL_RAMP_TIME: Time in soc_therm_clks to wait for DFLL throttling to engage

40.9.193 SOC_THERM_EDP_OC2_BBC_MAP_0

BBC Mapping

When the corresponding BBC_STS is received, assert the following BBC-dedicated OC-alarm

ALARM_A == OC2, ALARM_B == OC3, ALARM_C == OC4

Offset: 0x610 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0x0	BBC_STS_1111: mapping of BBC 1111->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
29:28	0x0	BBC_STS_1110: mapping of BBC 1110->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
27:26	0x0	BBC_STS_1101: mapping of BBC 1101->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
25:24	0x0	BBC_STS_1100: mapping of BBC 1100->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C

Bit	Reset	Description
23:22	0x0	BBC_STS_1011: mapping of BBC 1011->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
21:20	0x0	BBC_STS_1010: mapping of BBC 1010->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
19:18	0x0	BBC_STS_1001: mapping of BBC 1001->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
17:16	0x0	BBC_STS_1000: mapping of BBC 1000->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
15:14	0x0	BBC_STS_0111: mapping of BBC 0111->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
13:12	0x0	BBC_STS_0110: mapping of BBC 0110->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
11:10	0x0	BBC_STS_0101: mapping of BBC 0101->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
9:8	0x0	BBC_STS_0100: mapping of BBC 0100->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
7:6	0x0	BBC_STS_0011: mapping of BBC 0011->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
5:4	0x0	BBC_STS_0010: mapping of BBC 0010->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
3:2	0x0	BBC_STS_0001: mapping of BBC 0001->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C
1:0	0x0	BBC_STS_0000: mapping of BBC 0000->OC Alarm 0 = DISABLED 1 = ALARM_A 2 = ALARM_B 3 = ALARM_C

40.9.194 SOC_THERM_EDP_OC2_BBC_MAP_CTRL_0

Offset: 0x614 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MAPPING_ENABLE: enable bbc oc alarm mapping 0 = DISABLE 1 = ENABLE

40.9.195 SOC_THERM_THROTTLECTL_HVC_PSKIP_CTRL_0

Offset: 0x760 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.9.196 SOC_THERM_THROTTLECTL_HVC_PSKIP_RAMP_RATE_0

Offset: 0x764 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP:Pulse skip step size, 0:+/-1

40.9.197 SOC_THERM_THROTTLECTL_HVC_PSKIP_STATUS_0

Offset: 0x768 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:12	X	CURR_PULSE_SKIP_M:
11:4	X	CURR_PULSE_SKIP_N:
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

40.9.198 SOC_THERM_SIMON_AMASTER_ACCESS0_0

Offset: 0x900 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	ADDR

40.9.199 SOC_THERM_SIMON_AMASTER_DATA0_0

Offset: 0x904 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

40.10 Temperature Sensor Calibration Registers

40.10.1 FUSE_TSENSOR1_CALIB_0

Chip Option: tsensor1_calib

NVJTAG - temperature sensor calibration CP_TS_CPU2_offset=tsensor1_calib[12:0], FT_TS_CPU2_offset=tsensor1_calib[25:13]

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR1_CALIB

40.10.2 FUSE_TSENSOR2_CALIB_0

Chip Option: tsensor2_calib

NVJTAG - temperature sensor calibration CP_TS_CPU3_offset=tsensor2_calib[12:0], FT_TS_CPU3_offset=tsensor2_calib[25:13]

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR2_CALIB

40.10.3 FUSE_TSENSOR0_CALIB_0

Chip Option: tsensor0_calib

NVJTAG - temperature sensor calibration CP_TS_CPU1_offset=tsensor0_calib[12:0], FT_TS_CPU1_offset=tsensor0_calib[25:13]

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR0_CALIB

40.10.4 FUSE_TSENSOR3_CALIB_0

Chip Option: tsensor3_calib

NVJTAG - temperature sensor calibration CP_TS_CPU4_offset=tsensor3_calib[12:0], FT_TS_CPU4_offset=tsensor3_calib[25:13]

Offset: 0x22c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR3_CALIB

40.10.5 FUSE_TSENSOR4_CALIB_0

Chip Option: tsensor4_calib

NVJTAG - temperature sensor calibration CP_TS_GPU_offset=tsensor4_calib[12:0], FT_TS_GPU_offset=tsensor4_calib[25:13]

Offset: 0x254 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR4_CALIB

40.10.6 FUSE_TSENSOR5_CALIB_0

Chip Option: tsensor5_calib

NVJTAG - temperature sensor calibration

Offset: 0x258 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR5_CALIB

40.10.7 FUSE_TSENSOR6_CALIB_0

Chip Option: tsensor6_calib

NVJTAG - temperature sensor calibration CP_TS_MEM1_offset=tsensor6_calib[12:0], FT_TS_MEM1_offset=tsensor6_calib[25:13]

Offset: 0x25c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR6_CALIB

40.10.8 FUSE_TSENSOR7_CALIB_0

Chip Option: tsensor7_calib

NVJTAG - temperature sensor calibration CP_TS_PLLX_offset=tsensor7_calib[12:0], FT_TS_PLLX_offset=tsensor7_calib[25:13]

Offset: 0x260 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR7_CALIB

40.10.9 FUSE_TSENSOR_COMMON_0

Chip Option: tsensor common

NVJTAG - Information to calibrate the different thermal sensors that is not linked to a specific sensor

Offset: 0x280 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR_COMMON

40.10.10 FUSE_TSENSOR9_CALIB_0

Chip Option: tsensor9_calib

NVJTAG - temperature sensor TSOSC - PMC AO partition. Note that there is no tsensor8_calib for legacy reasons.

Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR9_CALIB

40.10.11 FUSE_TSENSOR10_CALIB_0

Chip Option: tsensor10_calib

NVJTAG - per-temperature sensor calibration, targeting an automotive application

Offset: 0x31c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	TSENSOR10_CALIB

40.10.12 FUSE_SPARE_REALIGNMENT_REG_0

Chip Option: spare_realignment_reg

NVJTAG - temperature sensor calibration

Offset: 0x37c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SPARE_REALIGNMENT_REG

CHAPTER 41: BOOT AND POWER MANAGEMENT PROCESSOR-LITE

41.1 Overview

The Tegra® X1 Boot and Power Management Processor is referred to as “Boot and Power Management Processor – Lite”, or BPMP-Lite, as it represents a first architectural step towards a dedicated boot and power management processor. Future Tegra devices will add further features, and more dedicated processing power, and will likely drop the “Lite” nomenclature when describing this processor.

The Boot and Power Management Processor – Lite, or BPMP-Lite, is an ARM7™ based microcontroller used for power management and boot purposes. It is similar to, and based on, the processor referred to as AVP or COP in prior Tegra devices, but no longer has any responsibility for audio or video decoding in Tegra.

Outside of boot, the BPMP-Lite runs the “BPMP Firmware” (BPMP-FW). This firmware provides various services related to runtime power management. To implement these services, the BPMP-FW “owns” particular hardware resources within the Tegra X1 processor. The CPU is never allowed to access such hardware directly. The BPMP-FW provides proxy services to give the CPU indirect access to such hardware where appropriate.

In cold boot, the BPMP-Lite processor runs the boot ROM code as well as a portion of the Boot Loader. As part of cold boot, the BPMP-FW is loaded into DRAM and the CPU triggers the BPMP-Lite processor to begin executing that firmware.

The BPMP-Lite processor handles low-level power management sequences and assists in various power state transitions.

41.1.1 Deep Sleep Support

The BPMP-Lite stays active after CPU complex power domain has been power gated or the VDD_CPU rail has been switched off. BPMP-Lite is the last standing processor on a system when it progressively transitions into lower power SOC Idle states. The system supports a mechanism of entering deep sleep (SC7 state) which is decoupled from the last standing CPU core executing a WFI instruction.

The BPMP-Lite processor plays the following roles in deep sleep state transitions:

- Manages system deep-sleep entry by correlating the CPU idle residency times (CC7 latency tolerance) to the energy cost of deep-sleep transitions and monitoring wake events and interrupts when the CPU complex is powered off.
- Performs deep-sleep entry housekeeping tasks on SOC resources
- Triggers deep-sleep entry without requiring the main CPU complex to be kept powered on.
- Performs warm boot steps on deep-sleep exit.

As reset is released, the BPMP-Lite wakes up and secures the system. The BPMP-Lite, which is not a TrustZone® master, then defers TrustZone configuration to the CPU. Upon wake-up by the BPMP-Lite, the CPU (which has defaulted to a secure state) configures the TZ-DRAM aperture registers in the Memory Controller in order to secure the TrustZone Execution Environment (TEE). Once TEE has been locked down by the CPU, the BPMP-Lite can initiate untrusted BPMP-Lite firmware.

Deep-sleep entry through the CC7 state is supported from any of the CPU cores in the cluster. Any of the four cores can be the last standing core in the cluster prior to CC7 or SC7 entry. CC7 to SC7 entry is mainly supported through the BPMP-Lite processor. See Chapter 16: CPU Complex in this TRM for more information on the CPU Power states.

41.2 Address Map

Refer to Chapter 2: Address Map of this document for the address map details.

41.3 Interrupts

Interrupts are controlled for the BPMP-Lite by the system interrupt controller. Refer to Chapter 3: Interrupt Controller of this document. All system interrupts are available to be routed to the BPMP-Lite.

41.4 IRAM and Crossbar Bus

IRAM is a block of 256 KB of on-chip ‘internal’ RAM assigned to the BPMP-Lite for code and data.

IRAM is configured as four contiguous banks of 64 Kilobytes of RAM. Each of these banks is accessible over a crossbar bus architecture, so that simultaneous access is possible to each bank from the crossbar masters.

The masters on the crossbar bus are:

- The ARM7 core
- The CPU complex over the AXI to ARM7 bridge
- The AHB bus interface

41.5 Chip-Level Features Involving BPMP

The BPMP-Lite participates in various chip level features. The following table shows the features where involvement of the BPMP is:

Table 245: BPMP Chip Features List

	Feature
CPU Power States Policies/Flows	
CPU Power States Policies/Flows	Retention/CC-state support
CPU Power States Policies/Flows	Fmin @ Vmin = HVC ? no role
CPU Power States Policies/Flows	Idle Aware EDP Management Mode
Intermediate Wake States	
Intermediate Wake States	SOC idle states
Frequency Management	
Frequency Management	DFS for BPMP-Lite
Clock Hardware Management	
Clock Hardware Management	MC/EMC PLL control

41.5.1 Cold Boot

The BPMP-Lite processor participates in the cold boot flow by running the boot ROM code and running (a portion of) a boot loader. As part of cold boot, the BPMP-FW is loaded into DRAM and the CPU triggers the BPMP-Lite processor to begin executing that firmware.

There are at least two relevant flavors of cold boot: boot from storage media and boot via “USB Recovery Mode”. Both involve the loading and execution of BPMP-FW. See Chapter 13: Boot Process for the cold boot flow.

41.5.2 CC-State Transitions

The various CC states (CPU Complex Power States) and the related state transitions are described in Chapter 18: Flow Controller. The BPMP-FW assists in the transitions to/from CC4, CC6, CC7, and the Cluster Switch.

41.5.3 SC-State Transitions

On Tegra X1, the SOC idle states have been enhanced. More low power states have been inserted between the ACTIVE state “SC0” and the Deep Sleep state “SC7” where the SOC rail is off.

The various SC-states (System on Chip Power States) are described below. The BPMP-FW takes a leading role in most of the SC-state transitions.

Various preconditions must be met before dropping down to any particular SC-state. For example, Tegra cannot drop to SC2 unless the CCPLEX is in CC4-retention or deeper and DMA-capable peripherals are configured such that they won't access DRAM. Similarly, Tegra cannot drop to SC7 unless the CCPLEX is in CC7 and other conditions are met.

The BPMP-FW is responsible for determining when all preconditions for entering an SC-state are met. When the preconditions are met, BPMP executes a sequence to transition into that state.

41.5.4 SC7 (also known as LP0)

SC7 is a particularly special SC state. The BPMP-FW is central to SC7 entry (also known as LP0 entry). However, it is all but absent during SC7->SC0 transitions (also known as LP0 exit or LP0 wake).

The BPMP-FW implements the final decision of whether to transition Tegra X1 to SC7 (LP0). As software on the last running CPU prepares for entry to CC6/CC7, it optionally configures itself to tolerate a further transition to LP0. It advertises that tolerance to the BPMP-lite. Once the CCPLEX is in CC7, the BPMP-FW waits for all other preconditions to CC7 entry to be met. If a wake event comes to the CPU prior to SC7-preconditions being met, the BPMP allows the CPU complex to wake. If preconditions are met prior to a wake event, the BPMP-FW executes the SC7 entry sequence.

The BPMP-Lite processor plays a central role to wake from SC7. Both the boot ROM and the SC7 resume vector run on the BPMP-Lite processor. The BPMP-FW does not resume execution until very late in the SC7 wake sequence.

41.5.5 BPMP-Lite Clock Frequency Management

BPMP-Lite might be the only processor alive for a long time and it needs to manage its own DFS and idle time.

There is no separate clock controls for BPMP-Lite. BPMP-Lite shares its clock with the SCLK, HCLK, and PCLK. The clock trunks are all over the chip. Thus, the S/H/PCLK frequency will be managed properly.

1. BPMP-Lite can change the S/H/PCLK clock frequency as long as it is within the floor & ceiling frequencies set by the CPU.
2. BPMP-Lite has ownership of the I2C5 when CCPLEX is in CC4-Ret or deeper idle state. During this time, it can manage the S/H/PCLK DVFS.

41.5.5.1 ACTMON

ACTMON has a monitor for BPMP-Lite (previously AVP) that counts the clock cycles when BPMP-Lite is in halt state.

41.5.6 Idle Aware EDP Management Mode

As a CPU core enters into C7 state, BPMP-Lite boosts the CPU clock. As a core in C7 state gets a wake interrupt, BPMP-Lite drops the CPU clock and then BPMP-Lite wakes up the CPU core.

41.5.7 Out of Scope Features

BPMP-Lite is not responsible for all power management. There are many power-management related features in which BPMP-Lite has no role.

41.6 BPMP-Lite Services

Table 246: Features and Services Mapping

	Chip Level Feature						
Service	**CC State assist**	**SC states assist**	**IAPM**	**MC Freq. Change**	**DRAM Temp Polling**	**S/H/P Clock Mgmt**	**ramDump**
Discovery	X	X	X	X	X		
Trace on							X

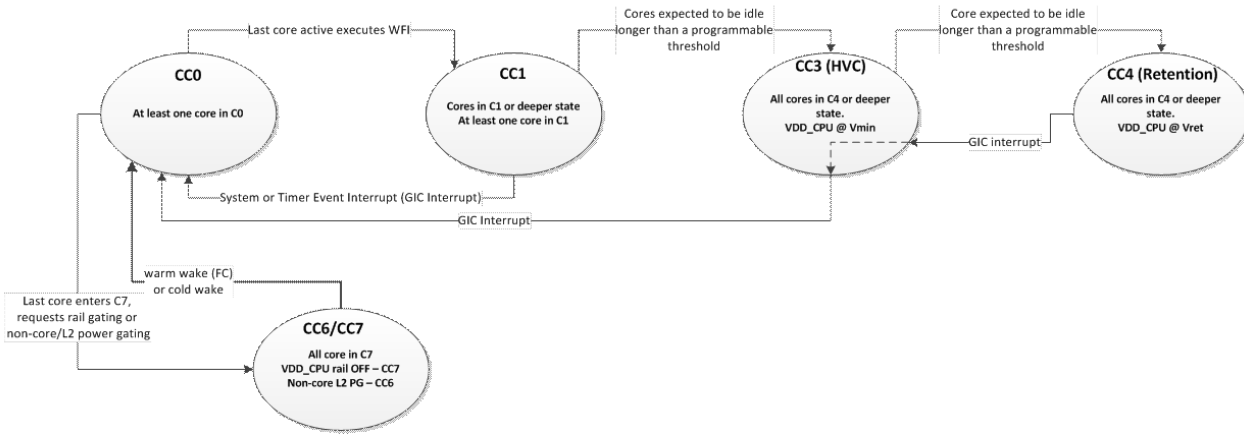
Table 246: Features and Services Mapping

	Chip Level Feature						
Service	**CC State assist**	**SC states assist**	**IAPM**	**MC Freq. Change**	**DRAM Temp Polling**	**S/H/P Clock Mgmt**	**ramDump**
CC4 Retention	X						
Calculate Deepest SC state		X					
Return to SC0		X					
CC7 Entry assist	X						
CC7 Exit assist	X						
SC7 Entry		X					
SC7 Exit		X					
SC2/3/4 Entry		X					
SC2/3/4 Exit		X					
S/H/P CLK Proxy						X	
WatchDog for CPU	X	X	X	X	X	X	
S/H/P Clock management						X	

41.6.1 CC States Support

Here is the top level of the CC states transition diagram:

Figure 172: CC States Transition



41.6.1.1 CC4-Retention Sequencing

When the CCPLEX is in CC3-HVC and BPMP-Lite gets an interrupt from flow controller, BPMP-Lite makes the decision either to enter into CC4-retention or stay in CC3-HVC. It also coordinates the sequencing to enter and exit from CC4-retention state with the flow controller.

41.6.1.2 CC7 Entry

During CC7 entry, BPMP-Lite plays a role in coordinating with the flow controller to make sure there is no race condition and all cores are power gated. See details in a later section.

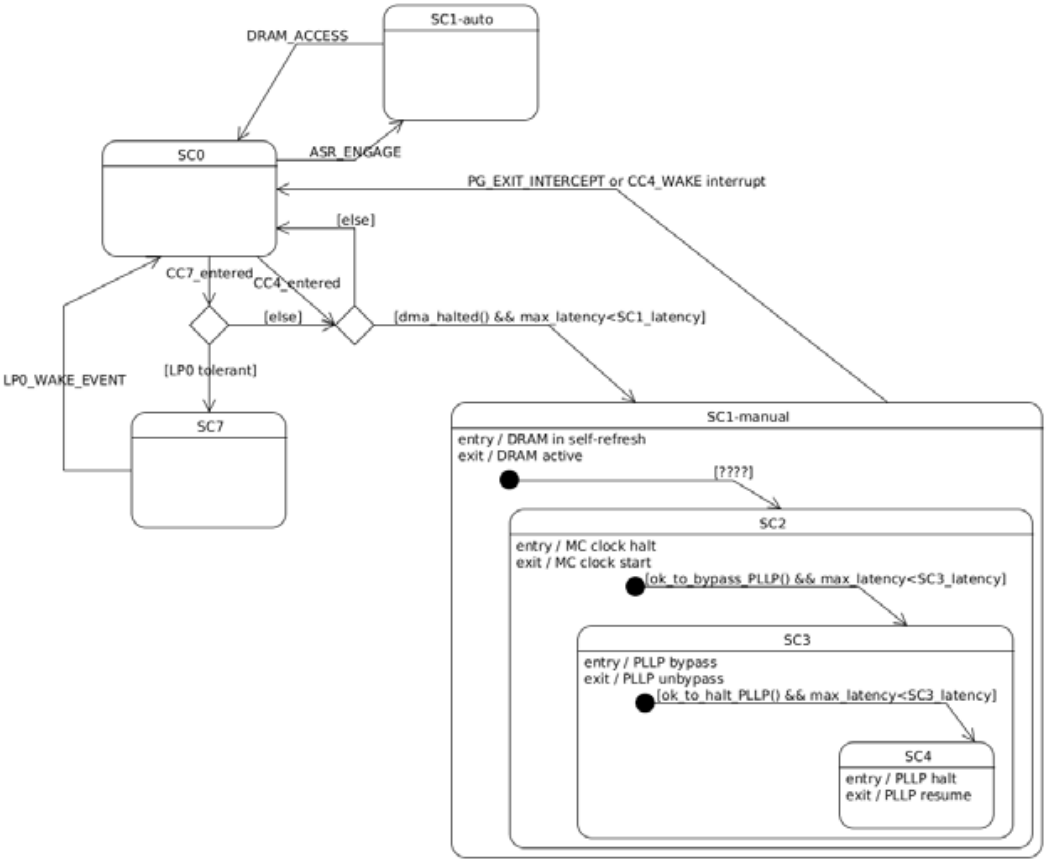
41.6.1.3 CC7 Exit

Upon SC7 exit, BPMP-Lite executes out of the Boot ROM, executes the (secure) SC7 exit vector, wakes the main CPU and then starts executing BPMP-Lite FW out of DRAM.

41.6.1.4 SC States Support

Refer to Section 41.9.3: SOC Idle States Support.

Figure 173: SC State Transition Diagram



41.6.1.5 SC7 Support

BPMP-Lite stays active after CPU complex power domain has been power gated or the VDD_CPU rail has been switched off. BPMP-Lite is the last standing processor on a system when it progressively transitions into lower power SOC Idle states. The system supports a mechanism of entering SC7 which is decoupled from the last standing CPU core executing a WFI instruction. Tegra X1 relies on BPMP-Lite in both ARM® based CPU complex and NVIDIA® Denver-based CPU complex.

The BPMP-Lite plays the following roles in SC7 state transitions:

1. Manages system SC7 entry by correlating the CPU idle residency times (CC7 latency tolerance) to the energy cost of SC7 transitions and monitoring wake events and interrupts when the CPU complex is powered off.
2. Performs SC7 entry house-keeping tasks on SOC resources
3. Triggers SC7 entry without requiring the main CPU complex to be kept powered on.
4. Performs warm boot steps on SC7 exit similar to previous generations of Tegra.

The BPMP is not capable of fully securing the Trust zone Execution Environment (TEE). In particular, the TZ-DRAM aperture registers in MC cannot be configured by the BPMP post SC7 exit. Therefore BPMP-FW is not run until the TEE is secured against any compromise by that code.

SC7 entry through CC7 state is supported from any of the CPU cores in the cluster. Any of the cores can be the last standing core in the cluster prior to CC7 or SC7 entry. On Tegra X1, CC7 to SC7 entry is mainly supported through BPMP-Lite.

The system extends the debug mode of SC7 entry using the PMC_DPD_ENABLE bit on previous generations to enable the BPMP-Lite to control SC7 entry. BPMP-Lite waits for CC7 entry to complete and then performs SC7 entry steps for the system resources including putting the DRAM in self refresh. Prior to triggering SC7 entry, BPMP-Lite ensures that there are no outstanding transactions in the system. It must disable its own interrupts. The flow controller must be prevented from triggering a CC7 exit to avoid race conditions. The last step performed by BPMP-Lite is to write to the PMC_DPD_ENABLE register and execute a halt. Following this, PMC hardware puts the system in SC7 state.

41.6.2 WatchDog for Main CPU Service

The software watchdog timer service for the main CPU implemented on BPMP-Lite allows for a graceful system shutdown after saving the state. Then it can reset and restart the system after which the state can be restored.

WatchDog Timer WDT4 can be dedicated to BPMP-Lite which can run slower than the main CPUs watchdog timers. If WDT4 expires, it is a sign of a hard CPU hang. When that happens, BPMP-Lite can log to CSITE or drive it on a UART. BPMP-Lite can also flush MC clients and put the DRAM into self-refresh.

41.7 BPMP-FW

The power management firmware (BPMP-FW) mainly resides in DRAM. It executes on the BPMP with run-time code being present either on DRAM or IRAM.

41.8 Hardware Resources

The hardware resources are split into five categories:

- Hardware resources owned by BPMP-FW
- Hardware resources accessible to BPMP-FW when main CPU is powered down
- Shared resources which are intrinsically safe
- Shared resources which require Locking
- Hardware resources not accessible by BPMP-FW

41.8.1 Hardware Resources Owned by BPMP-FW

This section lists the various hardware resources owned by the BPMP-FW. Code running on the CPU should never access these resources directly. The BPMP-FW provides proxy services for indirect access where appropriate.

- ARM7 CPU
- ARM7 cache
- IRAM[1]
  - BPMP-FW can choose to allocate particular addresses within IRAM for communication with CPU
- A DRAM carveout for BPMP-FW
- Some FC registers
  - FLOW_CTLR_FC_SEQUENCE_INTERCEPT_0
  - FLOW_CTLR_BPMP_CLUSTER_CONTROL_0
- CAR registers:
  - CLK_RST_CONTROLLER_SCLK_BURST_POLICY_0

1. There is an exception here for the Windows “crashdump” feature. When code on the CPU detects that the system is irretrievably hung, it can halt BPMP, write arbitrary data to IRAM, and trigger a full-system reset.

  - CLK_RST_CONTROLLER_SUPER_SCLK_DIVIDER_0
  - CLK_RST_CONTROLLER_CLK_SYSTEM_RATE_0
  - CLK_RST_CONTROLLER_COP_CLK_SKIP_POLICY_0
  - CLK_RST_CONTROLLER_CLK_SOURCE_SYS_0
  - CLK_RST_CONTROLLER_PLLMB_MISC1_0
- Some PMC registers
  - APBDEV_PMC_PWRGOOD_TIMER_0
  - APBDEV_PMC_CPUPWRGOOD_TIMER_0
  - APBDEV_PMC_CPUPWROFF_TIMER_0
  - APBDEV_PMC_WAKE_DELAY_0
- Some LIC registers
  - GIC_FWD_EN_BLOCK_BLOCK_CCPLEX_GIC_INTR_0
- Some interrupts
  - All interrupts are configurable to be routed to the main CPU or BPMP-Lite with the following exceptions:
    - WDT4-FIQ: is dedicated to the BPMP-Lite
    - Timer and software initiated interrupts inside the GIC are not routed to BPMP-Lite.
- SOC timers:
  - TMR2 (Tegra Timer 2)
- WatchDog timers:
  - WDT4

41.8.2 Hardware Resources Accessible to BPMP-FW When Main CPU is Powered Down

The following resources are time shared between main CPU and BPMP-Lite. BPMP-Lite accesses these resources only when the main CPU is in a power down state; otherwise these resources are owned by the main CPU when the main CPU cluster is in CC3-HVC state and above:

- I2C5 Control and Status Registers
- VDD_CORE voltage control
- Flow Controller
  - FLOW_CTLR_FC_SEQUENCE_INTERCEPT_0
  - FLOW_CTLR_CC4_FC_STATUS_0
  - FLOW_CTLR_CPU_PWR_CSR_0
- PMC
  - APBDEV_PMC_CNTRL2_0
  - APBDEV_PMC_WAKE_MASK_0
  - APBDEV_PMC_WAKE_LVL_0.
  - APBDEV_PMC_DPD_PADS_ORIDE_0
  - APBDEV_PMC_DPD_ENABLE_0
  - APBDEV_PMC_PLLP_WB0_OVERRIDE_0

41.8.3 Shared Resources Which are Intrinsically Safe

Resources in this section are shared by CPU and BPMP-FW at runtime. Access to the resources is intrinsically safe due to the design of the resource. For example, set/clear registers with certain bits owned by each processor are intrinsically safe.

- All LIC IER registers that have set and clear controls.

41.8.4 Shared Resources Which Require Locking

Resources in this section are shared by CPU and BPMP-FW at runtime and require some form of locking or mutual exclusion for the processors to make that sharing “safe”.

None exist in this class.

41.8.5 Hardware Resources Not Accessible by BPMP-FW

Resources not explicitly listed in the previous sections as allocated to, shared by, or time-shared by the BPMP-FW are forbidden to the BPMP-FW.

This list includes the following registers but not limited to:

- Flow Controller Reg: FLOW_CTLR_CC4_HVC_CONTROL_0
- Flow Controller Reg: FLOW_CTLR_CC4_RETENTION_CONTROL_0
- Flow Controller Reg: FLOW_CTLR_CC4_COREx_CTRL_0
- Flow Controller Reg: FLOW_CTLR_CC4_HVC_RETRY_0

41.9 Low Level Sequences

The following is a table of which sequence belong to which service that BPMP-Lite performs.

Table 247: Services and Sequences Mapping

	Services performed by BPMP-Lite								
Sequences	**Discovery**	**Trace on**	**SC4/7 Entry**	**CC4 Retention**	**SCLK Proxy**	**Calculate Deepest SC state**	**Return to SC0**	**CC7 Entry**	**CC7 Exit**
CC4-Ret				X					
CC7 Entry from any								X	
CC7 Entry from last								X	
CC7 Exit									X
SC7 Entry			X						
SC4 Entry			X						
SC4 exit							X		
Cluster Switch									

41.9.1 CC-States Assist

41.9.1.1 CC3–HVC□CC4-Retention and CC7 State

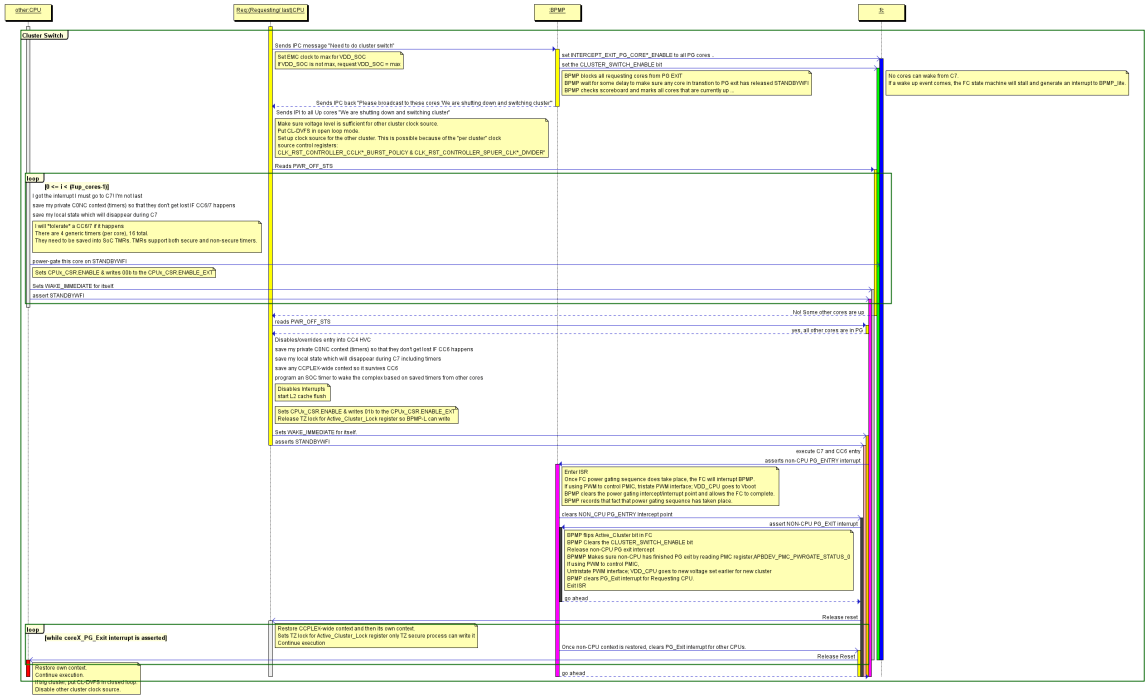
Refer to Chapter 18: Flow Controller for a description of these and their management.

41.9.2 Cluster Switch

The detailed Cluster Switch sequence is captured in the figure below.

BA is not supported for the slow cluster. To prevent spurious toggling, software must disable the BA_EDP logic and enable SLCG for the EDP logic before switching to the slow cluster. The reverse must be done before returning to the fast cluster.

Figure 174: Cluster Switch Sequence



41.9.3 SOC Idle States Support

41.9.3.1 SC2 Entry

SC2, SC3, and SC4 Entries are managed by the BPMP-FW. The main CPU sends a mailbox message to BPMP-Lite with information regarding tolerance for deeper idle SOC states based on runtime_PM (runtime power management) status.

The prerequisite for entering the SC2 state is that all DMA engines are turned off since the memory clock is turned off in this state.

41.9.3.2 SC3 and SC4 Entry

The prerequisite for entering the SC3 and SC4 states is the use of external thermal sensors. Since both SC3 and SC4 turn off the PLLP clock tree, SOC-Therm cannot run. Another prerequisite is that UART or any other modules need to use a non-PLLP originated clock.

In addition, the mailbox message from the main CPU to BPMP-Lite indicates tolerance to entering SC3 or SC4 based on the runtime_PM statuses that the main CPU checks before sending the message to BPMP-Lite.

41.9.3.3 SC2 SC3 and SC4 Exit

For SC2, SC3, and SC4 exits, no secure code is required to run since all states were saved prior to going into the deeper states.

41.9.3.4 SC7 Entry and Exit

Entry Procedure

The software processes involved in SC7 entry are complex. The Linux implementation of LP0 on Tegra SOCs involves the kernel, CPU Idle governor, PSCI interface and the EL3 Secure Monitor besides the BPMP-Lite FW that executes the chip-specific sequence for LP0 entry. The Android Linux kernel running on each individual CPU core signals the BPMP-Lite directly to indicate the deepest CPU cluster power state and SoC power states that the core can tolerate. The kernel does trigger CPU idle state entry via PSCI, but the core receives prior approval to enter certain CPU cluster and SoC idle states via BPMP interaction before entering the PSCI CPU_SUSPEND call. The EL3 secure monitor firmware does not communicate directly with the BPMP-Lite.

Exit Procedure

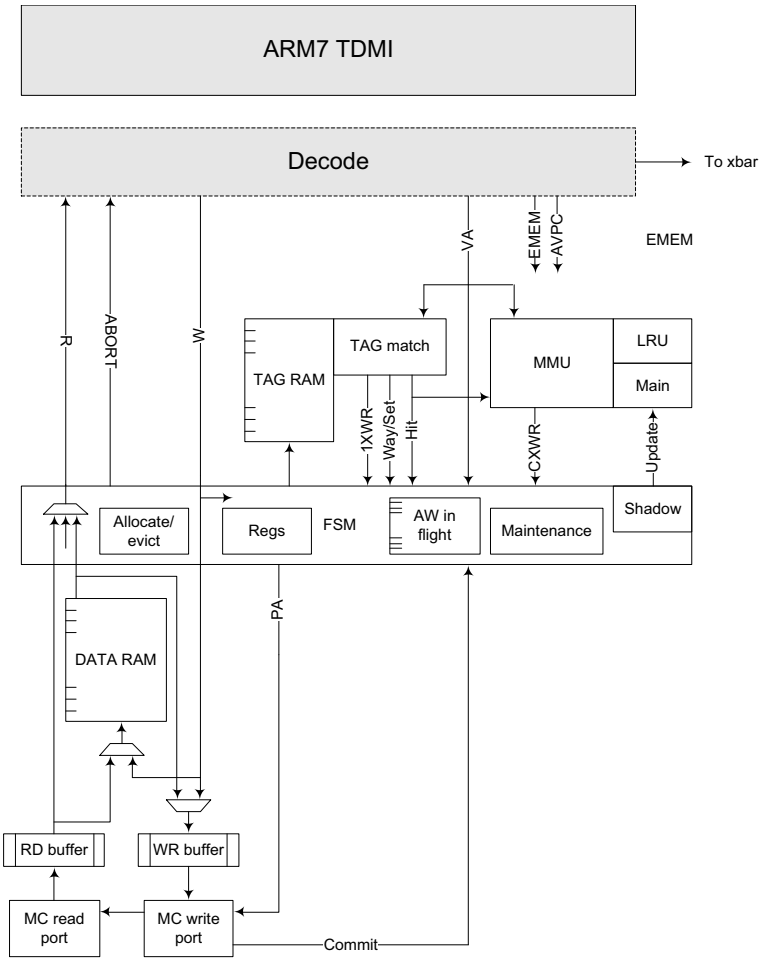
For SC7 exit, the BPMP-Lite SC7 resume code runs before the CCPLEX SC7 resume code runs. The BPMP-Lite SC7 resume code is responsible for booting the CCPLEX. The BPMP-Lite SC7 resume code runs from IRAM, after being copied over from DRAM to IRAM by Boot ROM and authenticated by Boot ROM. The BPMP-Lite SC7 resume code sequence is a separate entity from the BPMP-Lite firmware. The BPMP-Lite firmware is loaded by the CCPLEX late in SC7 exit.

41.10 Hardware Components

41.10.1 ARM7 and Cache

The following figure shows the block diagram of BPMP-Lite complex along with the cache.

Figure 175: BPMP-Lite Complex Block Diagram Showing the BPMP-Lite Cache



41.10.1.1 ARM7

ARM7 is a 32-bit RISC core based on the ARM v4 architecture that is designed for low power and small area applications. It is a three stage pipeline and it executes 32-bit ARM instruction set as well as 16-bit Thumb instruction set. The TDMI stands for:

- *T= Thumb instructions capable
- *D= Debug port
- *M = Fast multiply
- *I = ICE, in-chip emulator

41.10.1.2 BPMP-Lite (AVP) Cache

The BPMP-Lite cache is 32 Kbytes, 4-way set associative, unified (instruction & data). It is shown in Figure 175 above.

The BPMP-Lite does not have a full-fledged SMMU. It only adds attributes. So VA == PA.

41.10.2 IPC with CPU

The hardware synchronization primitives are used for inter-processor synchronization between the main CPU complex and the BPMP-Lite. Four types of mechanisms are provided for the synchronization.

- The shared semaphore provides a way to sync between the producer and consumer.
- Arbitrator semaphore is an arbitrator to decide if various resources are available for CPU or BPMP.
- Mailbox is like a single element FIFO between CPU and BPMP.
- Atomic: parallel to the existing Semaphores block, it is intended to extend that block's functionality with new capabilities. The advantage is to allow some specific algorithms that access shared resources to be performed lock free, for better performance.

The mailboxes and the Atomic resources can also be used as pointers to larger messages in iRAM and DRAM.

The shared mailboxes have the ability to generate interrupts in either direction on Full/Empty flags

Current mailboxes with the main CPU are implemented in hardware and they are only one deep in each direction.

The current hardware on Tegra X1 implements the mailbox in registers on the PPSB bus. Access to the hardware mailbox from BPMP is relatively fast (2-3 cycles). Reads to the hardware mailbox from main CPU are similar to reads from DRAM on the order of ~100 CPU cycles.

In addition, the software can implement software mailboxes between BPMP and main CPU in IRAM or DRAM by using circular buffers or linked lists. Circular buffers would require simpler software pointers than linked lists.

Table 248: IPC Registers

Group	Register	Description
Arres_sema	RES_SEMA (R)	Semaphore status
	SHRD_SMP_SET (W)	Set semaphore
	SHRD_SMP_CLR (W)	Clear semaphore
Mailbox	SHRD_INBOX (RW) (COP2CPU)	
	SHRD_OUTBOX (RW) (CPU2COP)	
Arb sem	SMP_GNT_ST (R)	Sem status
	SMP_GET (w)	Req sem
	SMP_PUT (w)	Clear sem
	SMP_REQ_ST (r)	Pending status
Atomics	AP0_TRIGGER	
	AP0_SETUP_V[] (rw)	
	AP0_SETUP_C[] (rw)	

Table 248: IPC Registers

Group	Register	Description
	AP0_RESULT[] (r)	

41.10.3 Semaphores and Mailboxes

Refer to Chapter 4: Semaphores of this document for more details.

41.11 BPMP-Lite Registers

Refer Section 1.4: Reading Register Tables in the Introduction chapter for the register table protocol as well as recommendations for accessing registers.

Arbitration for the various resources on the crossbar bus can be configured via the Arbitration Priority registers. Each crossbar master can be assigned a priority for each crossbar resource so that the resource controller can evaluate the latency needs for each master and complete the requests in the most efficient manner.

It is important that each master is assigned a different priority for each resource. If any two priorities are the same the hardware will override the settings with "safe" values, which may not be what the programmer has intended. So when updating the priorities, you should program all of the priority registers at the same time (one for each master).

41.11.1 ARB_PRIO_CPU_PRIORITY_0

Shared Resource Priority for CPU Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00040090 (0bxxxxx000000001xxx000000010010000)

Bit	Reset	Description
26:24	0x0	HOST1X: 000=Highest Priority, 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x0	APB: Access Privilege to APB Bus Registers 00=Highest Priority, 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x1	PSB: Access Privilege to PSB Registers 00=Highest Priority, 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x0	IROM0: 000=Highest Priority, 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x0	IRAM3: 000=Highest Priority, 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

Bit	Reset	Description
8:6	0x2	IRAM2: 000=Highest Priority, 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x2	IRAM1: 000=Highest Priority, 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x0	IRAM0: 000=Highest Priority, 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

41.11.2 ARB_PRIO_COP_PRIORITY_0

Shared Resource Priority for COP Register

Offset: 0x4 | Read/Write: R/W | Reset: 0x012024c2 (0bxxxxx001001000xxx010010011000010)

Bit	Reset	Description
26:24	0x1	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x2	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

Bit	Reset	Description
11:9	0x2	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x3	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x0	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x2	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

41.11.3 ARB_PRIO_VCP_PRIORITY_0

Shared Resource Priority for VCP Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x02201209 (0bxxxxx010001000xxx001001000001001)

Bit	Reset	Description
26:24	0x2	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST

Bit	Reset	Description
14:12	0x1	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x1	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x0	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x1	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x1	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

41.11.4 ARB_PRIO_DMA_PRIORITY_0

Shared Resource Priority for DMA Register

Offset: 0xc | Read/Write: R/W | Reset: 0x0320365b (0bxxxxx011001000xxx011011001011011)

Bit	Reset	Description
26:24	0x3	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST

Bit	Reset	Description
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x3	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x3	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x1	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x3	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x3	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

41.11.5 ARB_PRIO_UCQ_PRIORITY_0

Shared Resource Priority for UCQ Register

Offset: 0x14 | Read/Write: R/W | Reset: 0x04204924 (0bxxxxx100001000xxx100100100100100)

Bit	Reset	Description
26:24	0x4	HOST1X: 000=Highest Priority 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

Bit	Reset	Description
23:21	0x1	APB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
20:18	0x0	PSB: 11=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = LOWEST
14:12	0x4	IROM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
11:9	0x4	IRAM3: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
8:6	0x4	IRAM2: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
5:3	0x4	IRAM1: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST
2:0	0x4	IRAM0: 111=Lowest Priority 0 = HIGHEST 1 = PRIORITY1 2 = PRIORITY2 3 = PRIORITY3 4 = PRIORITY4 5 = PRIORITY5 6 = PRIORITY6 7 = LOWEST

CHAPTER 42: DESIGN FOR DEBUGGING (DFD)

The Tegra® X1 series processors leverage the enhancements in the design for debugging (DFD) feature. DFD enables the development and use of debugging tools that allow developers of Tegra-based implementations to place the Tegra X1 processor into known states and trace its behavior while running. Benefits of this enhanced DFD implementation include:

- Reduced power leakage
- Enhanced security
- Availability of standard interfaces

This chapter describes the Tegra X1 debuggable blocks and their debugging strategies. Use this information to help determine why something is not working in the software you have developed using NVIDIA® Tegra Board Support Package (BSP).

42.1 Related Documentation

Information in the following documentation is important to understand how to develop a debugging implementation using DFD. Some of these manuals are available from the public section of the ARM Infocenter website; others you may have to order from ARM.

- CoreSight™ SOC Revision: r2p1 Technical Reference Manual
  - http://infocenter.arm.com/help/topic/com.arm.doc.ddi0480d/DDI0480D_coresight_soc_r2p1_trm.pdf
  - Describes the ARM® CoreSight SOC 400 components. You must understand these component descriptions to be able to use the information in this document.
- CoreSight Trace Memory Controller Revision: r0p1 Technical Reference Manual
  - http://infocenter.arm.com/help/topic/com.arm.doc.ddi0461b/DDI0461B_tmc_r0p1_trm.pdf
  - Describes the Trace Memory Controller.
- CoreSight System Trace Macrocell Revision: r0p1 Technical Reference Manual
  - http://infocenter.arm.com/help/topic/com.arm.doc.ddi0444b/DDI0444B_stm_r0p1_trm.pdf
  - Describes the System Trace Macrocell.
- CoreSight Program Flow Trace PFTv1.0 and PFTv1.1 Architecture Specification
  - http://infocenter.arm.com/help/topic/com.arm.doc.ihi0035b/IHI0035B_cs_pft_v1_1_architecture_spec.pdf
  - Describes the ARM CoreSight Program Flow Trace architecture.
- ARM Cortex®-A57 MPCore Processor Revision: r1p0 Technical Reference Manual
  - Cortex_A57_MPCore_Technical_Reference_Manual_r1p0-00eac0.pdf
  - ARM Cortex-A57 Technical Reference Manual
- ARM Cortex-A53 MPCore Processor Revision: r0p1 Technical Reference Manual
  - DDI0500B_cortex_a53_r0p1_trm.pdf
  - ARM Cortex-A53 Technical Reference Manual
- ARM Architecture Reference Manual ARMv8
  - DDI0487A_a_armv8_arm.pdf
  - Reference information for ARMv8-A architecture profile.

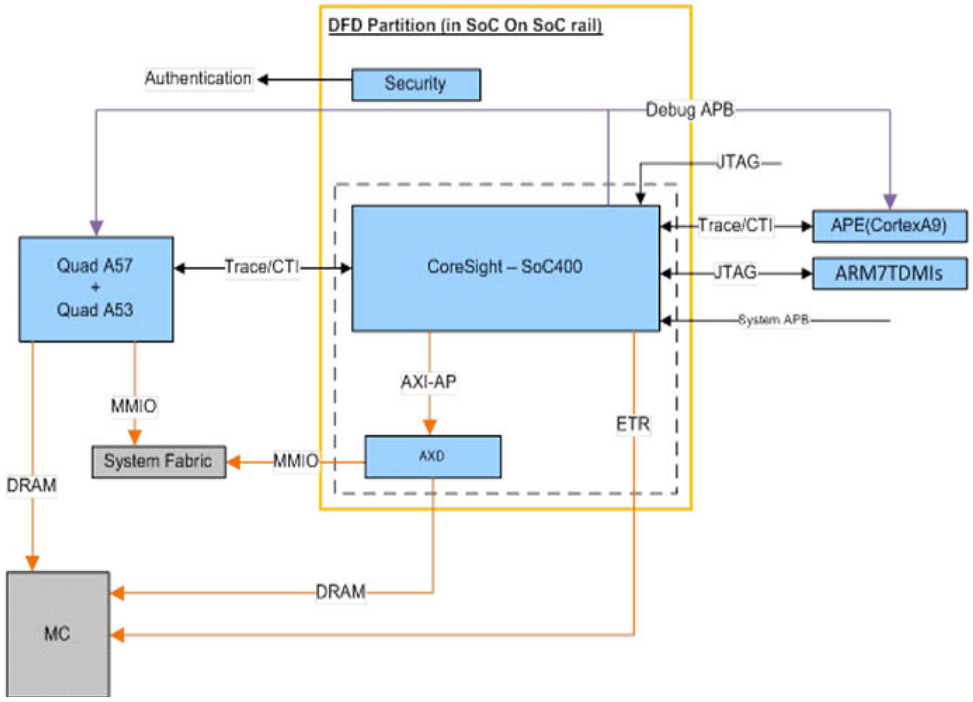
42.1.1 Tegra X1 DFD Components

Tegra X1 DFD components are contained in the CoreSight SOC-400 block. The location of this block, and interactions between it and other blocks, are described in this document.

42.1.2 Tegra X1 DFD Operation Overview

The following figure illustrates the general operation of the DFD components in Tegra X1.

Figure 176: DFD Components



Tegra X1 houses the DFD units in a separate power gateable partition shown in the yellow border in the diagram.

At the center of the block diagram is the CoreSight-SOC400 block. This block integrates with additional necessary IPs, such as the AXI address decoder (AXD), to interface with the SoC to yield the CoreSight Subsystem, indicated by the dotted border within the yellow border. Off chip access is realized via the standard Joint Test Action Group (JTAG) interface as in previous Tegra devices (IEEE 1149.1).

The CoreSight subsystem interfaces with the Cortex-A57/Cortex-A53 CCPLEX and provides the required CoreSight interfaces – Trace, Cross Trigger Matrix (CTM), Authentication, and the Debug Advanced Peripheral Bus (APB) – to provide all the architected debug support for the ARM v8 Cortex-A57/Cortex-A53 subsystem.

The CoreSight subsystem provides a similar debug ecosystem for the v7A based Cortex-A9 CPU called Audio DSP, within the Audio Processing Engine (APE).

For the v4 ARM7TDMI's Boot and Power Management Processor (BPMP), CoreSight provides only the JTAG interface as the ARMv4 architecture is older and not compatible to CoreSight.

CoreSight has complete access to the SoC registers (MMIO space). This, together with the Debug APB that can access the CPU registers, gives CoreSight complete access rights as the main Cortex-A57/Cortex-A53 CPUs. This is a major improvement in Tegra X1 over previous Tegra devices where CoreSight was an AHB master and limited to accessing SoC registers below the AHB fabric only.

CoreSight connects to the Memory Controller (MC) for both the Advanced eXtensible Interface Access Port (AXI-AP) and Embedded Trace Router (ETR) using NVIDIA specific IPs.

In addition, a security block has been added on the ungated SoC rail for driving authentication signals. Taken together, these new features provide Tegra X1 DFD unified CPU trace and SoC debug.

42.1.2.1 Managing CPU Power Down During Debug

Normal power management can interfere with the ability to debug CPUs. To allow debug, CPU power-down can be placed into an emulation mode so that the normal controls do not actually power down the CPUs. Refer to the section titled 'DBGNOPWRDWN' in the Flow Controller chapter of this TRM.

42.2 Tegra X1 DFD Improvements

The following table identifies DFD improvements in Tegra X1.

Table 249: Tegra X1 DFD Improvements

Hardware DFD Feature	Tegra K1 32 Bit	Tegra X1	Improvements
Connection to CPU (JTAG)	Debug Communication Channel	Debug Communication Channel with Memory Access Mode in v8	Faster code download/upload via the debugger.
Connection to AXI-AP (JTAG)	AHB-AP, 32 bit address. Can only access AHB peripherals	AXI-AP 34 bit address, Can access complete MMIO and DRAM without requiring SMMU	Complete system access when CPUs are powered down, dead, under reset, clock disabled, or unable break in.
Connection to ARM7(JTAG)	JTAG TAP	JTAG TAP	No change.
Connection to APE (JTAG)	N/A	CoreSight	All regular debug functionality is available to the APE that is available to the main CPU.
Trace Storage	ETB: 8KB circular buffer	ETF: 16KB circular buffer	Larger buffer for longer duration trace. Preserved through WDT resets.
Trace Streaming to DRAM	Not available	ETR: descriptor based throughput matched DMA to stream ETF to DRAM	Allows to get data output on WDT boot via USB or WiFi. Dramatically increases trace storage (1 MB with provision to increase as desired).
Trace Pipeline	32 bit	128 bit	No data drops when tracing at full CPU frequency.
SW Instrumentation (UART, USB)	CCPLEX UART	CCPLEX UART, APE UART, STM	STM can combine printfs from various threads and stream to DRAM and out over USB for closed system logging.
Debug APB Deadcodes	CCPLEX only	CCPLEX, APE	Provides information on accessing power gated, clock gated, reset CPU, and results in timeouts for other cases.
Power gateable DFD partition	DFD logic was all over the SoC	Separate Power-gateable partition	Continuous power savings on production systems where debug functionality is not required.
DFD security	BCT, Boot ROM, boot loader, NV legacy	BCT and Boot ROM only	Simplified and standardized
DFD Reset	Multiple standard resets which gets asserted on WDT.	Single Debug reset that does not assert on WDT reset	CoreSight ETF pointers preserved on WDT reset.
Cache Flush on WDT FIQ	Not supported	Supported	Debug of Soft-Lockups that require latest contents in DRAM
Scan Dump (JTAG)	Scan Dump GUI with web based utility to parse scanned dumps	Scan Dump GUI with web based utility to parse scanned dumps	No change.

42.2.1 CoreSight Major and Minor Blocks

The CoreSight SoC-400 module is divided into CoreSight Major and CoreSight Minor blocks.

The CoreSight Major block contains:

- The main Advanced Peripheral Bus (APB) Interconnect
- Funnel

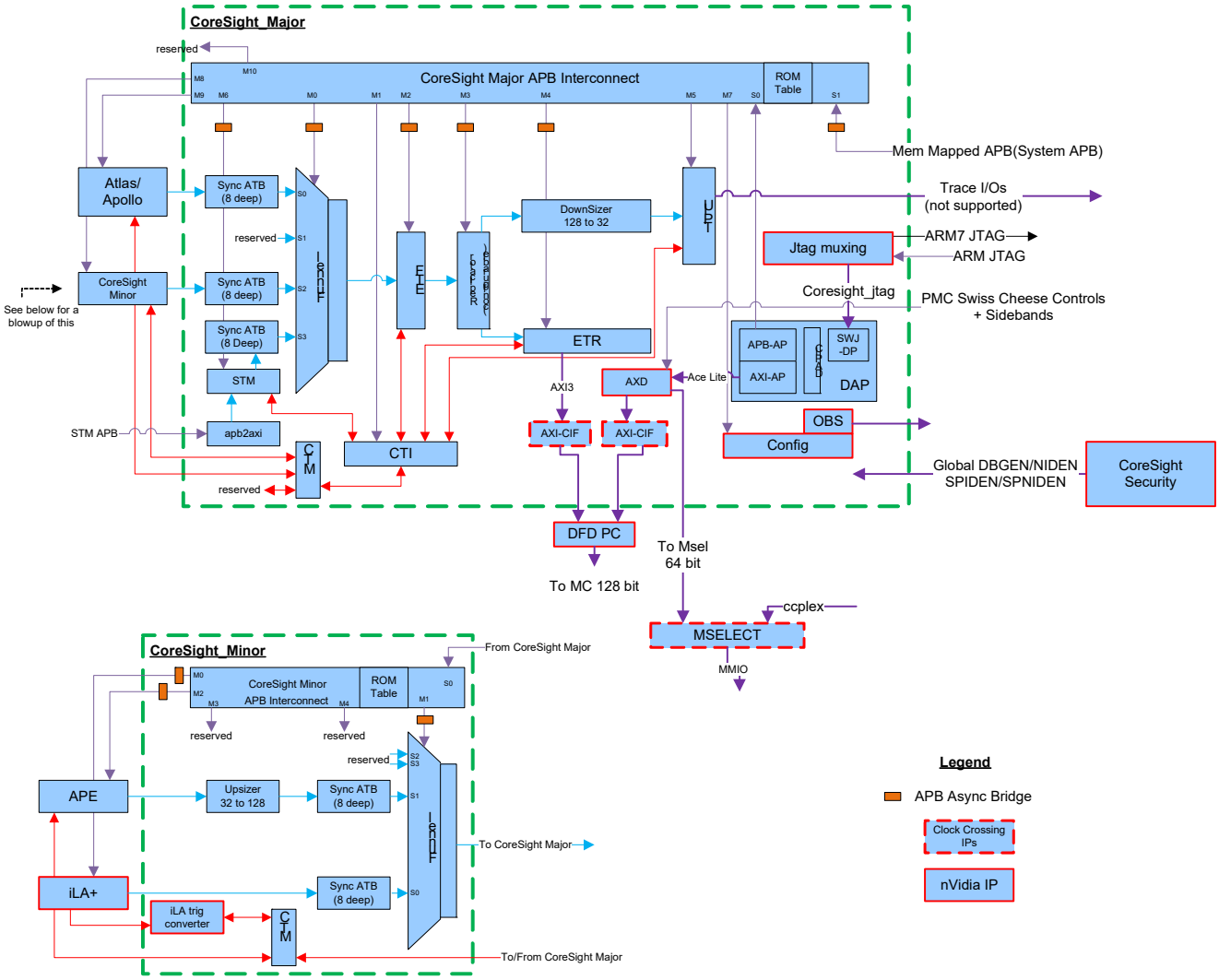
- Embedded Trace FIFO (ETF)
- Replicator
- Embedded Trace Router (ETR)
- Cross-Trigger Interface (CTI)
- Cross-Trigger Matrix (CTM)
- System Trace Macrocell (STM)
- Trace Port Interface Unit (TPIU)
- Bridges (for holding data while the funnel arbitrates across trace sources)
- APB Async Bridges (AB) to decouple Debug APB clock from the ATB clock

The CoreSight Minor block contains:

- A staggered APBIC (slave to the CoreSight_Major APBIC)
- Funnel (slave of the CoreSight_Minor APBIC)
- Sync ATB bridges
- CTM

The following figure illustrates the CoreSight microarchitecture.

Figure 177: CoreSight Major and Minor Block Diagram



42.2.2 Tegra X1 DFD CoreSight Components

The following table shows the CoreSight components that implement Tegra X1 DFD. A synopsis of the DFD functionality provided for each component.

Table 250: CoreSight Components with DFD

Component	Acronym	Synopsis
APB Interconnect	APBIC	Connects to APB-AP and System APB on the slave ports and provides master APB ports for all DFD components in the CoreSight subsystem, in various CPUs, and an NVIDIA specific configuration aperture. Occupies 32 MB on the system AMAP.
Funnel	N/A	Arbitrates traces from multiple sources and provides a single Trace bus.
Embedded Trace FIFO	ETF	16-kilobyte (KB) embedded buffer organized as 256 x 512 (<width> x <depth>) to store trace data.
Embedded Trace Router	ETR	Streams ETF contents to off-chip DRAM via an AXI master interface. Capable of streaming to secure DRAM apertures. Not capable of reading VPR region.
Upsizer (32 to 128)	N/A	Upsizes the trace bus by sampling across 4 clock cycles and transmitting every fourth cycle for matching throughput requirements.
Downsizer (128 to 32)	N/A	Downsizes the 128-bit trace interface into 32 bits to connect to TPIU.
Replicator	N/A	Replicates a single trace interface into 2 trace interfaces.
Trace Port Interface Unit	TPIU	Streams trace data to off-chip interface.Trace IOs are not brought out on Tegra X1. This module is not used but added for future use.

Table 250: CoreSight Components with DFD

Component	Acronym	Synopsis
Debug Access Port	DAP	Contains the AXI-AP, APBAP, SWJDP and DAPIC.
AXI-Access Port	AXI-AP	Converts JTAG into a master AXI interface to connect to system fabric. Capable of making secure accesses into the system. Not capable of reading VPR region.
APB Access Port	APBAP	Converts JTAG into debug APB master interface. Provides master interface to connect.
DAP Interconnect	DAPIC	Interconnect within the DAP to extend the JTAG interface to the APB AP and AXI-AP.
Serial Wire JTAG Debug Port	SWJDP	Interfaces with the external JTAG port and connects to the DAPIC. Serial Wire (reduced pin) not supported in Tegra X1. Only JTAG is supported.
Cross Trigger Matrix	CTM	Transmits events across trace sources.
Cross Trigger Interface	CTI	Connects to various trigger interfaces.
System Trace Macrocell	STM	Collects SW instrumentation (printfs) from system and coverts that data into ATB.
Timestamp Interpolator	TS INTP	Takes a slow timestamp and increases timestamp granularity.
APB Sync Bridge	N/A	APB pipe macro to close timing on the Debug APB interface.
APB Async Bridge	N/A	APB bridge for clock crossing.
ATB Sync Bridge	N/A	ATB pipe macro to close timing on ATB interface.
ATB Async Bridge	N/A	ATB bridge for clock crossing.

42.2.3 CoreSight System Interfaces

The following table shows the CoreSight system interfaces for Tegra X1 DFD, the components accessed for each interface and the address and data width for each interface.

Table 251: CoreSight System Interfaces for DFD

Interface	Accesses	Address Width	Data Width
Debug APB	CPU Components: (Cortex-A57 and Cortex-A9): PTM, CTI, PMU, and debug registers. CoreSight Components: APBIC, Funnel, ETF, Replicator, ETR, STM, TPIU, CTI This bus can have two masters: 1. JTAG from debugger for external debugging. 2. System APB for self-hosted debugging (debug code running on CPU). **Note**: ATB and APB sync/async bridges, Interpolator, Upsizer/ Downsizer, and CTM do not have any configurable registers and hence do not interface with Debug APB.	Variable, depends on the APB slave. But always less than 25.	32
System APB	CCPLEX and ARM7TDMIs can master the Debug APB using the System APB bus. This path is useful for self-hosted debugs. Also used for interfacing to the STM via the apb2axi bridge (32-bit data)	25	32
AXD AXI to System Fabric Registers (AXD + AXIAP)	Use this bus when it is not possible to halt or connect to the CPU due to a CPU hang/powerdown. Also use this bus for real-time debug (without disturbing the CPU). The actual physical address is set up by the debugger in the AXI-AP Control Status Word register to emanate an access into the possible AMAP of the SoC. This interface supports secure and non-Secure accesses. This interface is originally Ace-Lite from AXI-AP but reduced to AXI3 to connect to AXD.	34	64
MC interface (AXD + ETR)	The DRAM arm from AXD (34-bit address and 64-bit data) is combined with the AXI ETR traffic (34-bit address and 128-bit data) and sent to MC. This interface allows you to query/modify any DRAM location via the AXI-AP and stream trace contents via the ETR.	34	128
Trace Interface	One-trace Interface from the CCPLEX, One-trace Interface from the APE. The trace Interface does not have an address bus.	N/A	128 (CCPLEX) 32(APE)
Trigger Interface	One event and acknowledgment CTM interface from the CCPLEX, One event and acknowledgment CTM interface from the APE.	N/A	16

Table 251: CoreSight System Interfaces for DFD

Interface	Accesses	Address Width	Data Width
Authentication	Sidebands from the CoreSight configuration block consisting of global dbgen, niden, spniden, and spiden, for use by the Cortex-A9 and the Cortex-A57/Cortex-A53 processors and the internal CoreSight modules.	N/A	7
JTAG	This interface is used to connect to the external debugging equipment and is an implementation of the standard IEEE1149.1 specification. The TRST pin is not connected from the debugger, it is connected to the same power on reset. TRST is repurposed by NVIDIA to select signals to send JTAG chip, I/Os to DFT controller (TRST=1, NV_JTAG), or CoreSight (TRST=0, ARM_JTAG) The JTAG interface in Tegra X1 is daisy-chained across the ARM7TDMIs TAP and DAP TAP and is called Serial Mode: JTAG TDI --> ARM7TDMIs TDI. ARM7TDMIs TDO --> CoreSight TDI. CoreSight TDO --> JTAG TDO.	N/A	1 bit (TDI or TDO)

42.2.4 CoreSight Logic in CCPLEX

Earlier Tegra devices exported a 32-bit ATB interface for each CPU to CoreSight. This approach is not scalable as the number of CPUs change or as the number of CPU clusters change. Also, a 32-bit interface cannot handle the ATB throughput. Consequently, the CCPLEX adds another staggered interconnect with a funnel to combine the quad CPU trace with adequate buffering for the funnel switching times.

42.3 Implementing Tegra X1 DFD Components

The following sections provide information for implementing DFD on specific Tegra X1 components.

42.3.1 Audio Processor Engine (APE)

The Tegra X1 APE uses the ARM Cortex-A9 processor. The Cortex-A9 includes a discrete Program Trace Macrocell (PTM) instanced in the CPU power domain by the APE. The APE interfaces to the CoreSight Minor block for the APBIC, Funnel, and CTM. The trace bus from APE is 32-bits wide and upsized in the CoreSight Minor block.

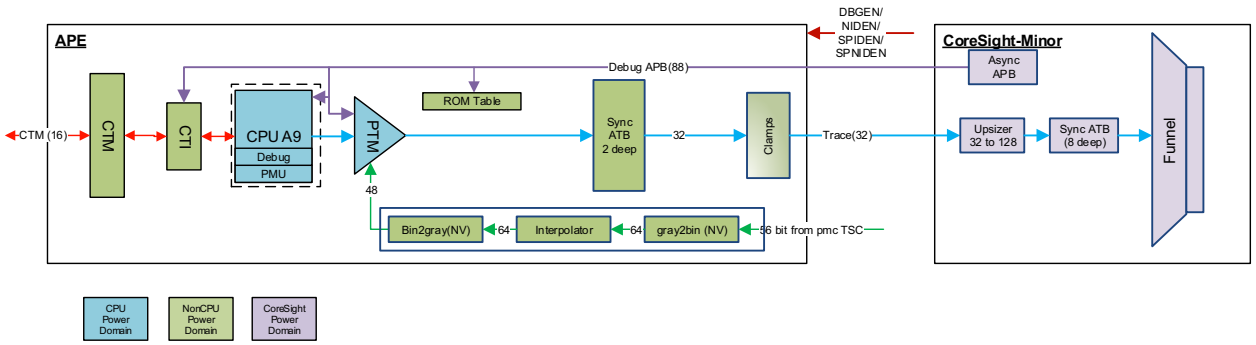
42.3.2 CoreSight Connections to the Audio Processor Engine

Tegra X1 has an Audio Processor Engine (APE) that uses a Cortex-A9 processor. The Cortex-A9 processor has a discrete PTM that is instanced in the CPU power domain by the APE.

The APE interfaces to CoreSight Minor for the APBIC, Funnel, and CTM. The trace bus from the APE is 32 bits wide and upsized in the CoreSight Minor block.

The following figure illustrates the APE CoreSight connectivity.

Figure 178: APE to CoreSight Connectivity



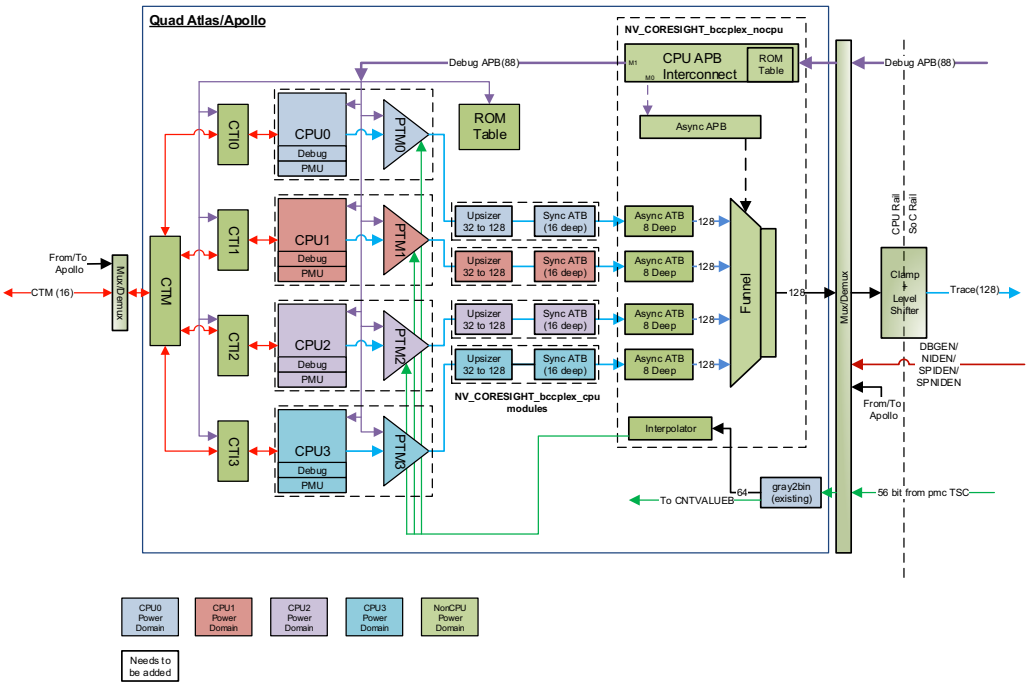
42.3.3 Cortex-A57/Cortex-A53 4+4 Switched Cluster Implementation

Tegra X1 is a quad Cortex-A57 and a quad Cortex-A53 implementation. Both Cortex-A57 and Cortex-A53 are implementations of the ARMv8 architecture and support identical debug feature list. Either the Cortex-A57 or Cortex-A53 can be active at a given time based on a register bit setting in the Flow Controller. The interfaces from CoreSight do not change at all with this switched cluster implementation. The CCPLEX accepts the interfaces from CoreSight and guarantees that these are correctly routed to the currently active cluster. The cluster switch happens only in CC6 state where both the clusters are off. The same set of registers from CAR is used to control the resets and clock enables for both the clusters. Therefore, CoreSight RTL implementation assumes that there is only one cluster. This also allows the debugger to identify a single set of ROM tables, either for the Cortex-A53 or for Cortex-A57.

42.3.4 Quad Cortex-A57/Cortex-A53 CoreSight Connectivity

The following figure illustrates the connections between the quad CPUs and CoreSight.

Figure 179: Quad CPUs and CoreSight Connectivity



42.4 CoreSight ROM Tables

Each APBIC implements a ROM table which tabulates the configurable components hanging off the APBIC. The ROM tables are housed at offset 0 of the defined aperture and are automatically read off by debuggers to determine the debug architecture of the SoC. There are:

- Three ROM tables were added by Tegra X1 DFD: one each in CoreSight Major and CoreSight Minor, and a third in the CoreSight non-CPU block
- One ROM table each in the CCPLEX complex
- One ROM table in the APE for a total of five ROM tables.

The CoreSight_Major ROM table points to:

- CoreSight Major Components
- CoreSight Minor ROM table (first level nested ROM)

- CoreSight Big Interconnect ROM table (first level nested ROM)

The CoreSight Minor ROM table points to the APE ROM table (second-level nested ROM).

The CoreSight Big Interconnect ROM table points to:

- CCPLEX ROM table (second level nested ROM)
- CCPLEX Funnel

The CCPLEX ROM table points to:

- Four PTMs
- Four CTIs
- Four PMUs
- Four DBGs (for either Cortex-A57 or Cortex-A53)

The APE ROM table points to:

- One PTM
- One CTI
- One PMU
- One DBG

Note: *The ROM in the ROM tables does not indicate a physical read-only memory. The ROM table is static data based on defines in the APBIC.*

42.5 CoreSight System Trace Macrocell

For software instrumentation, CoreSight implements a System Trace Macrocell. This is the ARM replacement for the older, narrower, slower Instrumentation Trace Macrocell. STM has a 32-bit address and 32-bit data AXI slave port to receive software instrumentation. The AMAP stipulated by STM is 16 MB.

42.6 CoreSight Address Map

The following table shows the CoreSight address map for Tegra X1 DFD. The table shows the base, start, and end addresses for each component, and the APBIC master port if available

Table 252: CoreSight Address Map

CoreSight Major			
Interface	**Start Address**	**End Address**	**APBIC Master Port**
CoreSight Major Interconnect ROM	7200_0000	7200_FFFF	Internal to Top APBIC
Funnel	7201_0000	7201_FFFF	M0
CTI	7202_0000	7202_FFFF	M1
ETF	7203_0000	7203_FFFF	M2
Replicator	7204_0000	7204_FFFF	M3
ETR	7205_0000	7205_FFFF	M4
TPIU	7206_0000	7206_FFFF	M5
STM Debug APB	7207_0000	7207_FFFF	M6
Reserved	7208_0000	727E_FFFF	Reserved
CoreSight NVIDIA specific Aperture	727F_0000	727F_FFFF	M7
CoreSight Minor Interconnect	7280_0000	72FF_FFFF	M8
CPU Big Interconnect	7300_0000	727F_FFFF	M9

Table 252: CoreSight Address Map

Reserved	7380_0000	73FF_FFFF	Reserved
CoreSight Minor Interconnect			
Component	**Start Address**	**End Address**	**APBIC Master Port**
CoreSight Minor Interconnect ROM	7280_0000	7280_FFFF	Internal to Sub APBIC
Reserved	7281_0000	7281_FFFF	M0
Funnel	7282_0000	7282_FFFF	M1
Reserved	7283_0000	729F_FFFF	-
APE	72A0_0000	72BF_FFFF	M2
Reserved	72C0_0000	72DF_FFFF	M3
Reserved	72E0_0000	727F_FFFF	M4
CPU Big Interconnect			
Component	**Start Address**	**End Address**	**APBIC Master Port**
CPU Big Interconnect ROM	7300_0000	7300_FFFF	Internal to Sub APBIC
CPU Big Funnel	7301_0000	7301_FFFF	M0
CPU Big ROM	7340_0000	7340_FFFF	M1
CPU0_DBG	7341_0000	7341_FFFF	-
CPU0_CTI	7342_0000	7342_FFFF	-
CPU0_PMU	7343_0000	7343_FFFF	-
CPU0_ETM	7344_0000	7344_FFFF	-
CPU1_DBG	7351_0000	7351_FFFF	-
CPU1_CTI	7352_0000	7352_FFFF	-
CPU1_PMU	7353_0000	7353_FFFF	-
CPU1_ETM	7354_0000	7354_FFFF	-
CPU2_DBG	7361_0000	7361_FFFF	-
CPU2_CTI	7362_0000	7362_FFFF	-
CPU2_PMU	7363_0000	7363_FFFF	-
CPU2_ETM	7364_0000	7364_FFFF	-
CPU3_DBG	7371_0000	7371_FFFF	-
CPU3_CTI	7372_0000	7372_FFFF	-
CPU3_PMU	7373_0000	7373_FFFF	-
CPU3_ETM	7374_0000	7374_FFFF	-
STM			
STM AXI Registers	7100_0000	71FF_FFFF	

42.7 ETM/PTM Overview

Each ARM processor in Tegra X1 integrates either the Embedded Trace Macrocell (in the ARM Cortex-A57 and ARM Cortex-A53 processors) or the Program Trace Macrocell (in the ARM Cortex A9 processor) – referred to as the Trace Module. As the CPU executes instructions, it also outputs waypoint information (change in PC value) to the Trace Module. The Trace Module forwards these waypoints, post-packetization and compression, according to the supported trace protocol (ETMv4 for the Cortex-A57 and Cortex-A53 processors, or PFT1.0 for the Cortex-A9 processors), to CoreSight ETF along with a Trace ID, using the AMBA Trace Bus protocol (ATB). A formatter in ETF injects the Trace IDs along with the data packets and stores them in a circular 16KB buffer.

This 16 KB buffer is read out and the data is segregated based on IDs. Subsequently the segregated data is decompressed and depacketized for a parser to determine the exact waypoints. The parser also has a copy of the source code and is able to map the exact sequence of instructions that a CPU executed.

Getting the exact sequence of instructions executed is very beneficial in the case of a hard lockup. Other uses for the trace include code optimization for easy determination if some portion of code is causing additional latency.

42.8 CoreSight ATB Interface Throughput Calculations

One CPU generates three bits of instruction trace (non-cycle accurate) for 99.5 percent trace, and six bits for the cycle accurate case. CoreSight ATB sizing is done based on the non-cycle accurate tracing. The following table identifies the traces generated per CPU.

Table 253: Bits per Cycle per CPU

CPUs	Bits Per Cycle
1 Cortex-A57 CPU	3 bits per cycle
Quad Cortex-A57 CPUs	3 x 4 bits per cycle
Quad Cortex-A57 CPUs	3 x 4 x1.25 bits per cycle (v8 scaling factor is 1.25, as provided by ARM)
1 Cortex-A57 CPU	3 bits per cycle

The number of bits generated in one second is equal to 3 x 4 x 1.25 x <CPU_frequency>.

For CoreSight running at maximum CPU/4 frequency, the ATB interface width *N*, with protocol efficiency of 80% throughput handled is:

$$N \times \frac{CPU\_Frequency}{4} \times 0.8$$

This must be able to handle the CPU trace output running at CPU frequency, i.e.,

$$3 \times 4 \times 1.25 \times CPU_{Frequency} = N \times \frac{CPU_{Frequency}}{4} \times 0.8$$

$$N = 75 \text{ bits}$$

Since ATB supports a width of 128 bits as the next highest after 64 bits, the ATB interface is 128-bit for Tegra X1 devices.

The increased data width protects from additional trace from APE in rare cases where trace must be collected simultaneously with the CPUs. This also provides trace for the cycle accurate case with reduced frequency from the quad Cortex-A57 CPUs at:

$$128 \times CoreSight_{Frequency} \times 0.8 = 6 \times 4 \times 1.25 \times CPU_{Frequency}$$

Where:

$$CPU_{Frequency} : CoreSight_{Frequency} = 3.4 : 1$$

The ATB interface width cannot be further increased because all CoreSight components peak at 128 bits.

42.9 CoreSight Trace Sinks ETF and ETR

The following table shows the CoreSight trace sink characteristics for Tegra X1 DFD, the corresponding Embedded Trace FIFO (ETF), ETR, and USB limits.

Table 254: CoreSight Trace Sink Characteristics

Characteristic	ETF (16 KB)	DDR via ETR DMA	USB
Throughput	52 Gbps @ 408 MHz and 128 bit (contact NVIDIA for higher frequency requirements.)	52Gbps	Real-time processor tracing requires reduction of CPU frequency.
Intrusive	No	Yes	Yes
Available on Commercial Devices	Yes	Yes	Yes

Table 254: CoreSight Trace Sink Characteristics

Characteristic	ETF (16 KB)	DDR via ETR DMA	USB
Use Cases	Collect trace for watchdog reset, and code optimization for the CCPLEX.	Collects trace information for watchdog reset, and code optimization for the CCPLEX (note the high bandwidth requirement at DDR = 25%).	Use for single CPU trace at low frequency, or APE only trace to avoid saturation of DRAM bandwidth Tracing limited to USB speeds. Use for single CPU trace at low frequency, or APE only trace to avoid saturation of DRAM bandwidth.

42.10 CoreSight AMBA Trace ID (ATID) Mapping

The following table shows the mapping for CoreSight AMBA Trace ID (ATID). When collecting trace from multiple sources, the trace sinks (ETF and ETR) use ATIDs to segregate trace data.

Table 255: CoreSight ATID Mapping

BCCPLEX (also called Fast Cluster or Big Cluster) using Cortex-A57 processors		
ATID	**Processor**	**Protocol**
0x40	CPU0	ETMv4
0x41	CPU1	ETMv4
0x42	CPU2	ETMv4
0x43	CPU3	ETMv4
LCCPLEX (also called Slow Cluster or Little Cluster) using Cortex-A53 processors		
ATID	**Processor**	**Protocol**
0x30	CPU0	ETMv4
0x31	CPU1	ETMv4
0x32	CPU2	ETMv4
0x33	CPU3	ETMv4
APE Cortex-A9		
ATID	**Processor**	**Protocol**
0x20	CPU0	PFT1.0
STM		
ATID	**Processor**	**Protocol**
0x10	NA	MIPI STP

42.10.1 CoreSight Clocks

The following table shows the CoreSight clocks, the maximum frequency for each clock, and where they are used.

Table 256: CoreSight Clock Information

Clock	Maximum Frequency	Used By
CoreSight ATB Clock	624 MHz. Recommend to run at PLLP of 408 MHz fixed clock across voltage corners	CCPLEX and APE traces
CoreSight Debug APB Clock	136 MHz	All debug APB logic
JTAG TCK	Adaptive – uses RTCK	CoreSight and ARM7TDMIsTDMIs TAP

42.11 CoreSight Resets

For debugging Watchdog Timer (WDT) reset scenarios, CoreSight survives the WDT reset that occurs on 4th expiration of WDT. This allows ETF pointers to be read after a WDT reset and recover the trace (survived due to no power cycle).

CoreSight comes out of reset by default without any software intervention.

42.12 CoreSight CTI Connections

The Cross Trigger Interface collects triggers from the ETF, ETR and STM.

This CTI is distinct from the CTI for each CPU that exists together with the PTM for each CPU. The CPU CTIs implement triggers for entering Debug, triggers for PTMs, and other triggers.

The CTIs across CoreSight, CCPLEX and APE communicate via the Cross Trigger Matrix (CTM). The CTM receives the CTI triggers and converts them into channel events.

The following table shows the CoreSight connections with CTI input and output triggers.

Table 257: CoreSight CTI Connections

Input Trigger	Name	Driving Module
CTITRIGIN0	Full	ETF
CTITRIGIN1	ACQCOMP	ETF
CTITRIGIN2	Full	ETR
CTITRIGIN3	ACQCOMP	ETR
CTITRIGIN4	ASYNCOUT	STM
CTITRIGIN5	SPTE	STM
CTITRIGIN6	SW	STM
CTITRIGIN7	Tied off	NA
Output Trigger	**Name**	**Driving Module**
CTITRIGOUT0	TRIGIN	ETF
CTITRIGOUT1	FLUSHIN	ETF
CTITRIGOUT2	TRIGIN	ETR
CTITRIGOUT3	FLUSHIN	ETR
CTITRIGOUT4	TRIGIN	TPIU
CTITRIGOUT5	FLUSHIN	TPIU
CTITRIGOUT6	Not Connected	NA
CTITRIGOUT7	Not Connected	NA

For similar trigger connectivity for CTIs of the CPUs, refer to the respective CPU TRMs. For connectivity of CPU CTIs, refer to the respective CPU/PTM TRMs.

42.13 Timestamp Halt on Debug

The debug acknowledgments from the CCPLEX and ARM7TDMIs are used to halt the timers. This helps to debug the timer software for these modules and disable unwarranted WDT timeouts.

> ***Note:*** *APE DBGACK does not participate in this halt-on debug mechanism.*

42.14 CoreSight Debug APB Timeout and Access Block Deadcodes

In order to not hang the debugger while making accesses on the Debug APB, CoreSight adds a new internal NVIDIA module to perform the following functions:

- Time-limit the Debug APB accesses for the CCPLEX and the APE
- Block the accesses which are guaranteed to result in timeout due to a non-responsive CCPLEX or APE.

Either of these could be non-responsive due to power down, clock gating or reset.

In the event of either a timeout or access block, the access from CoreSight APB-AP completes by asserting respective master pready of the APBIC as 1. In addition, deadcodes are returned on the prdata bus to immediately identify what caused a

particular access to timeout or block. On receiving the deadcode, the debugger can query the respective CORESIGHT_CFG_<module>_ALIVE_STATUS register for the cause of timeout/blocking.

> **Note:** *In the event a CPU gclk register access operation results in a timeout, subsequent CPU pclkdbg register access operations also result in timeouts. The debugger must first fix the issue indicated by the ALIVE status register and then make subsequent access operations for the other registers.*

42.15 Lauterbach Debugger Configuration Parameters

CoreSight is a design kit with configurable components. CoreSight architecture must be indicated to external debuggers. ARM debuggers like DS5/DSTREAM boxes can detect this configuration by reading the registers within each core, already knowing the Instruction Register Lengths for various ARM CPUs.

Lauterbach, which caters to other processors outside the ARM ecosystem, requires Access Port (AP) configurations and Instruction Register Length for the additional TAP ports.

42.15.1 DAP

The CoreSight DAP has two access ports and each has a unique Identification Register (IDR). The Lauterbach debugger settings for the above DAP are listed in the following table.

Access Port	Identification Register (IDR)
APB-AP at port 0	0x44770002
AXI-AP at port 1	0x24770002

The DAP access port Lauterbach debugger settings are listed in the following table.

Access Port	Identification Register (IDR)
0x0	0x44770002 (MEM-AP APB) --> SYStem.CONFIG DEBUGACCESSPORT 0
0x1	0x44770001 (MEM-AP AHB) --> SYStem.CONFIG MEMORYACCESSPORT 1

42.15.2 TAP Position

The DAP is not the only Test Access Port (TAP) in the JTAG scan chain of Tegra X1 devices. You must set up the position of the ARM7TDMIsTDMIs TAP the JTAG scan chain:

```
TDI -> ARM7TDMIs -> DAP -> TDO
```

resulting in:

```
TAP1 = ARM7TDMIs, Instruction Register (IR) Length = 4
TAP2 = DAP
Data Register (DR) length is 1 bit for each TAP (IR = BYPASS)
SYStem.CONFIG.DAPIRPRE 0x0 ; IR for TAP after DAP - there is none.
SYStem.CONFIG.DAPIRPOST 0x4 ; IR TAP1
SYStem.CONFIG.DAPDRPRE 0x0 ; DR for the TAP after DAP - there is non
SYStem.CONFIG.DAPDRPOST 0x1 ; DR TAP1
```

42.15.3 Base Addresses

Refer to Section 42.6: CoreSight Address Map for any additional base address required. Define or otherwise code the offsets as the OR of the base of the module offset and 0x8000_0000, that is, with bit 31 set:

<Module_Offset> | 0x8000_0000 = <Final_Address>

For example, for the APE Debug Register, the offset is:

```
0xA10000 | 0x8000_0000 = 0x80A1_0000
```

42.16 DFD Partition

Tegra X1 devices group all debug units into a separate DFD partition that can be power gated.

The Tegra X1 DFD is a separately power-gateable partition of the device. Power to the DFD partition can be optionally applied during device boot, controlled by the boot loader. This allows production Tegra X1 devices to consume less power than previous NVIDIA devices because the boot loader can be built to not apply power to the DFD partition when debugging features are not needed.

To enable this power-saving feature, use the POWERGATE_DEBUG condition in the boot loader code to apply power to the DFD partition only for a debugging version of the software.

42.16.1 Possible CoreSight Accesses During Low Power States

When the CPU rail is powered down, the CoreSight AXI-AP port is used for debugging access. This requires that the DFD partition itself is not powered down in Suspend (LP1) and Deep Sleep (LP0) Debug states. The low power state access rights are listed in the following table.

Table 258: Access Rights in Low Power

Low Power State	Definition of the Low Power State	Access Rights
Deep Sleep (LP0)	DRAM is put in self-refresh, system state is saved in PMC + DRAM, VDD_SOC and VDD_CPU rails are powered off, and PMC is configured to monitor Deep Sleep (LP0) wake events which trigger Deep Sleep (LP0) exit.	No accesses can emanate from CoreSight as CoreSight which resides on the VDD_SOC rail is also powered down.
Suspend (LP1)	DRAM is put in self-refresh, VDD_CPU rail is powered off, devices are power-gated, SoC clock domains are put into min frequency (32Khz) and the flow-controller is configured to monitor Suspend (LP1) wake events which would trigger Suspend (LP1) exit.	CoreSight is clock enabled by default. It has osc clock as source and lives in the non-PG partition (AHB). Debugger remains connected via the AO JTAG pads. Daisy chain with ARM7TDMIs work as sclk is running. With DRAM is in self refresh, memory cannot be accessed. Registers of powered-up and clock-enabled modules can be accessed via the CoreSight AXI-AP port.

42.17 DFD Security

Debugging in ODM production Tegra devices can be disabled by:

- Blowing debug fuses. Recommended if debug never needs to be re-enabled on ODM-Production device. Each Debug Authentication signals has a corresponding fuse. Blowing fuse disables (zeroes out) the corresponding Debug Signal and the functionality governed by it.

Fuse Name	Debug Signal
FUSE_ODM_INFO[12]	DBGEN
FUSE_ODM_INFO[11]	NIDEN
FUSE_ODM_INFO[10]	SPIDEN
FUSE_ODM_INFO[9]	SPNIDEN
FUSE_ODM_INFO[8]	DEVICEEN
FUSE_ARM_JTAG_DIS	JTAG_ENABLE

- Not blowing the debug fuses and instead using a per-chip secure signed BCT. Recommended if there is a need to re-enable debug on ODM-Production devices.

  Debug for this mode is re-enabled by updating the value of the SecureDebugControl entry in the Boot ROM BCT.

Note: *For devices that are re-enabled for debug, it is strongly recommended that the JTAG interface pins: TMS, TCK, TDI, TDO, TRST, and RTCK are available as contact points on the main board. This allows partially opening the device and attaching a JTAG header.*

42.17.1 Re-enabling Debugging Using Per Chip Secure Signed BCT

To re-enable debugging, a unique Boot Configuration Table (BCT) is required per device. A BCT for one device does not enable debugging on another device. The per chip device unique ID allows you to apply control per chip BCT.

Prerequisites:

- NvBootECID: Specifies the Unique ID / ECID of the chip that the BCT is specifically generated for. This field is required if any debugging features are enabled through SecureDebugControl. Otherwise, this field is not required. This design prevents a signed BCT, with the debugging features enabled, from leaking into the field that would enable various debugging features for all devices signed with the same RSA private key or AES key.
- SecureDebugControl: Specifies which debugging features are enabled or disabled. This setting maps directly to APBDEV_PMC_DEBUG_AUTHENTICATION_0. It must specify the ECID of the chip in UniqueChipID on production systems.

Table 259: DFD Security BCT Entries Mappings

Name	Definition	Common Use Case(s)
DBGEN SecureDebugControl[5]	NonSecure Invasive Debug Enable	Must be 1 for CPUs to halt, AXI-AP to make system accesses, and for ETR to stream trace to DRAM.
NIDEN SecureDebugControl[4]	NonSecure NonInvasive Debug Enable	Must be 1 for PTM trace from CPUs.
SPIDEN SecureDebugControl[3]	Secure Invasive Debug Enable	Must be 1 for AXI-AP to make secure accesses into the system, ETR to write to Secure DRAM.
SPNIDEN SecureDebugControl[2]	Secure NonInvasive Debug Enable	Must be 1 for accessing secure registers in PMU and CPUs over the Debug APB.
DEVICEEN SecureDebugControl[1]	Device Enable	Must be 1 for accessing any registers on mapped over the Debug APB.
JTAG_ENABLE SecureDebugControl[0]	Enables JTAG TAP	Must be 1 to enable the JTAG interface to CoreSight-SoC400.

Boot ROM securely writes SecureDebugControl to APBDEV_PMC_DEBUG_AUTHENTICATION_0 as per table below.

Table 260: DFD Security Modes

Mode	APBDEV_PMC_DEBUG_AUTHENTICATION Handling	
	Tegra K1 32 Bit	Tegra X1
NvProduction and Secure Provisioning disabled	JTAG always enabled at exit	JTAG always enabled at exit
NvProduction and Secure Provisioning enabled	N/A	Depends on BCT*
ODM Production Mode	Depends on BCT†*	Depends on BCT†*
Notes		
* Check if the ECID specified in the BCT matches the ECID in the chip. The ECID check is an additional step for debug controls enablement purposes only. This is to ensure that a signed debug controls-enabled build can only work on one system. † For Tegra X1, the SecureDebugControl field in the BCT maps directly to DBGEN/NIDEN/ SPIDEN/SPNIDEN/DEVICEEN/ JTAG_ENABLE bits in APBDEV_PMC_DEBUG_AUTHENTICATION_0. For Tegra K1 32 Bit, the SecureJtagControl field controls this functionality.		

42.17.1.1 To Obtain a Target ECID of the Device

1. Place the device in recovery mode.

2. On the Linux host machine, open a terminal window to check if the device is listed by entering the lsusb command.

   The system returns a listing similar to the following:

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 0409:005a NEC Corp. HighSpeed Hub
Bus 008 Device 002: ID 046d:c05a Logitech, Inc. Optical Mouse M90
Bus 001 Device 003: ID 0403:6011 Future Technology Devices International, Ltd FT4232H Quad HS USB-UART/FIFO IC
Bus 001 Device 021: ID 0955:7021 NVidia Corp.
```

3. Execute the following command to obtain the UID of the device.

```
sudo $TOP/out/host/linux-x86/bin/tegrarcm –uid
```

   The system returns a 128-bit UID which must be interpreted as follows to obtain a 100-bit ECID:

```
BR_CID: 0x621010015c1971862000000002058300
MSB: 0x6 -> indicates the operating mode.0x6 stands for odm secure PKC mode.
rcm version : 0x2101 -> this indicates rcm version (0x21 -chipid  0x01 -minor revision)
ECID_0 = 0x02058300; // first 4bytes from LSB
ECID_1 = 0x20000000; // next 4bytes after ECID_0
ECID_2 = 0x5c197186; // next 4bytes after ECID_1
ECID_3 = 0x00000001; // ((next 4 bytes after ECID_2) & 0xF)
```

42.17.2 Generating the BCT with UID and SecureDebugControl

To generate the BCT with the UID, you must append the BCT configuration file with the parameters that were returned from the tegrarcm command in the previous topic. For example:

```
ECID_0=0x02058300;
ECID_1=0x20000000;
ECID_2=0x5c197186;
ECID_3=0x00000001;
SecureDebugControl=0x3f; // Value as per requirement
```

42.17.3 Signing the New BCT

Signing the BCT is performed as part of the process of generating the signed binaries for the secure device. You must obtain a private key for generating signed images.

42.18 CoreSight Registers

42.18.1 CORESIGHT_ETF_HUGO_CXTMC_REGS_LAR_0

This is used to enable write access to device registers.

Offset: 0x30fb0 I Read/Write: WO I Reset: 0xc5acce55 (0b11000101101011001100111001010101)

Bit	Reset	Description
31:0	-978530731	ACCESS_W:A write of 0xC5ACCE55 enables further write access to this device. A write of any value other than 0xC5ACCE55 will have the effect of removing write access. 3316436565 = Unlock

42.18.2 CORESIGHT_ETR_HUGO_CXTMC_REGS_LAR_0

This is used to enable write access to device registers.

Offset: 0x50fb0 I Read/Write: WO I Reset: 0xc5acce55 (0b11000101101011001100111001010101)

Bit	Reset	Description
31:0	978530731	ACCESS_W:A write of 0xC5ACCE55 enables further write access to this device. A write of any value other than 0xC5ACCE55 will have the effect of removing write access. 3316436565 = Unlock

42.18.3 CORESIGHT_ETF_HUGO_CXTMC_REGS_FFCR_0

This register allows user control of the stop, trigger, and flush events.

Offset: 0x30304 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000x000x000xx00)

Bit	R/W	Reset	Description
14	WO	0x0	DrainBuffer: This bit is used to enable draining of the trace data through the ATB Master interface after the formatter has stopped. This is useful in Circular buffer mode to capture trace data into trace memory and then drain the captured trace through the ATB Master interface. Writing a 1 to this bit when TMCReady=1 and TraceCaptEn=1 starts the drain of the contents of the trace buffer through the ATB Master interface. This bit always reads as zero. The TMCReady bit (STS register, 0x00C) goes low while the drain is in progress.This bit is functional only when the TMC is in Circular Buffer mode and formatting is enabled (EnFt bit in FFCR register is set). Setting this bit when the TMC is in any other mode or when formatting is disabled will result in Unpredictable behavior.Setting this bit other than when the TMCReady bit (STS Register, 0x00C) is set and TraceCaptEn=1 will result in Unpredictable behavior. Once trace capture is complete in Circular Buffer mode, all of the captured trace must be retrieved from the trace memory through the same mechanism - either read all trace data out through RRD reads or drain all trace data by setting the DrainBuffer bit. Setting the DrainBuffer bit after some of the captured trace has been read out through RRD will result in Unpredictable behavior. 1 = HIGH 0 = LOW
13	RW	0x0	StopOnTrigEvt: If this bit is set, the formatter is stopped when a Trigger Event has been observed. A Trigger Event is said to have occurred when the Formatter has written the set number of datawords (as programmed in the TRG register, 0x01C) into the trace memory after the occurrence of either a rising edge on the TRIGIN input or a trigger packet on the incoming trace stream (ATID = 7h7D). This bit is cleared on reset (disabled). Enabling the TMC in Software-read-FIFO mode or Hardware-read-FIFO mode with this bit set will result in Unpredictable behavior because in FIFO modes, the TMC is a trace link rather than a trace sink and trigger events are related to trace sink functionality. 1 = HIGH 0 = LOW
12	RW	0x0	StopOnFl: If this bit is set, the formatter is stopped on completion of a flush operation. The initiation of a flush operation is controlled by programming the register bits FlushMan, FOnTrigEvt and FOnFlIn in the FFCR register, 0x304. When a flush-initiation condition occurs, AFVALIDS is pulled HIGH. Once AFREADYS is sampled HIGH, trace capture is stopped. Any remaining data in the formatter is appended with a post-amble and written to trace memory. The flush operation is then said to have completed. This bit is cleared on reset (disabled).In the FIFO modes, if a flush is initiated due to any condition other than a manual flush request, its completion does not lead to a formatter stop regardless of the value programmed in this bit. When the TMC is configured as an ETF, if a flush is initiated by the ATB Master interface, its completion does not lead to a formatter stop regardless of the value programmed in this bit. 1 = HIGH 0 = LOW

Bit	R/W	Reset	Description
10	RW	0x0	TrigOnFI: = 0), then trigger indication on the trace stream is blocked regardless of the value programmed in this bit.When the TMC is configured as an ETF, if a flush is initiated by the ATB Master interface, its completion does not lead to a trigger indication on the trace stream regardless of the value programmed in this bit. 1 = HIGH 0 = LOW
9	RW	0x0	TrigOnTrigEvt: = 0), then trigger indication on the trace stream is blocked regardless of the value programmed in this bit. This bit is not supported in Software-read-FIFO mode or Hardware-read-FIFO mode because in FIFO modes, the TMC is a trace link rather than a trace sink and trigger events are related to trace sink functionality. 1 = HIGH 0 = LOW
8	RW	0x0	TrigOnTrigIn: = 0), then trigger indication on the trace stream is blocked regardless of the value programmed in this bit. 1 = HIGH 0 = LOW
6	WO	0x0	FlushMan_W: Manually generate a flush of the system. Setting this bit causes a flush to be generated. If TraceCaptEn bit in CTL register is 0, then writes to this bit are ignored. In Circular buffer mode and Hardware-FIFO mode, this bit is cleared automatically when AFREADYS is sampled HIGH. In Software-FIFO mode, this bit is cleared when the flush data is written to trace memory. This bit is clear on reset. 1 = HIGH 0 = LOW
6	RO	X	FlushMan_R: Manually generate a flush of the system. Setting this bit causes a flush to be generated. If TraceCaptEn bit in CTL register is 0, then writes to this bit are ignored. In Circular buffer mode and Hardware-FIFO mode, this bit is cleared automatically when AFREADYS is sampled HIGH. In Software-FIFO mode, this bit is cleared when the flush data is written to trace memory. This bit is clear on reset. 1 = HIGH 0 = LOW
5	RW	0x0	FOnTrigEvt: Setting this bit generates a flush when a Trigger event occurs. A trigger event is said to have occurred when the Formatter has written the set number of datawords (as programmed in the TRG register) into the trace memory after the occurrence of either a rising edge on the TRIGIN input or a trigger packet in the incoming trace stream (ATID = 7h7D). This bit is clear on reset. This bit is not supported in Software-read-FIFO mode or Hardware-read-FIFO mode because in FIFO mode, the TMC is a trace link rather than a trace sink and trigger events are related to trace sink functionality. 1 = HIGH 0 = LOW
4	RW	0x0	FOnFIIn: Setting this bit enables the detection of transitions on the FLUSHIN input by the TMC. If this bit is set and the Formatter has not already stopped, a rising edge on FLUSHIN initiates a flush request. This bit is clear on reset. 1 = HIGH 0 = LOW
1	RW	0x0	EnTI: Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 8h00 with ATID 7h7D in the trace stream. Trigger indication on the trace stream is additionally controlled by the register bits TrigOnFI, TrigOnTrigEvt and TrigOnTrigIn in the FFCR register, 0x304. This bit can only be changed when TMCReady=1 and TraceCaptEn=0. This bit takes effect only when the EnFt register bit in this register is set. If EnTI bit is set to 1 when EnFt is 0, it results in formatting being enabled. This bit is clear on reset. 1 = Trigger_insertion_enabled 0 = Trigger_insertion_disabled
0	RW	0x0	EnFt: ace data. Disabling of formatting is deprecated and is supported in the TMC for backwards-compatibility with earlier versions of the ETB. Hence, disabling of formatting is supported only in Circular Buffer mode. Features in the TMC such as the FIFO modes and the DrainBuffer bit that are not part of the earlier versions of the ETB do not support disabling of formatting. If EnTI bit is set to 1 when EnFt is 0, it results in formatting being enabled. If the TMC is enabled in a mode other than Circular Buffer mode with EnFt 0, it will result in formatting being enabled. Attempting to write to this bit when TMCReady=0 or TraceCaptEn=1 will result in Unpredictable behavior. 1 = HIGH 0 = LOW

42.18.4 CORESIGHT_ETF_HUGO_CXTMC_REGS_RSZ_0

Defines the size, in 32-bit words, of the local RAM buffer.

Offset: 0x30004 I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:0	X	RSZ: Size of the RAM in 32-bit words e.g., for 1KB RAM, this register field returns 0x00000100 on reads; for 4GB RAM, this register field returns 0x40000000 on reads.

42.18.5 CORESIGHT_ETF_HUGO_CXTMC_REGS_STS_0

TMC Status register

Offset: 0x3000c I Read/Write: RO I Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	Empty:-Buffer mode, it is possible that the Empty bit and the Full bit (STS register) are 1 at the same time due to the fact that the Full bit in this mode, once set, does not clear unless TraceCaptEn is set. 1 = HIGH 0 = LOW
3	X	FtEmpty: This bit is set when trace capture has stopped, and all internal pipelines and buffers have drained. Unlike TMCReady, it is not affected by buffer drains and AXI accesses. The ACQCOMP output reflects the value of this bit unless the TMC is in integration mode. 1 = HIGH 0 = LOW
2	X	TMCReady: This bit is set when all the following are true:Trace capture has stopped and all internal pipelines and buffers have drained. This is equivalent to being in the Stopped or Disabled state.The TMC is not training because of the DrainBuffer bit of the FFCR being set.In ETR configuration, the AXI interface is not busy. This case can be used to detect page table reads in scatter-gather mode when in Stopped state. 1 = HIGH 0 = LOW
1	X	Triggered: The Triggered bit is set when trace capture is in progress and the TMC has detected a trigger event. A trigger event is said to have occurred when the TMC has written the set number of datawords (as programmed in the TRG register) into the trace memory after the occurrence of either a rising edge on the TRIGIN input or a trigger packet (ATID = 7h7D) in the incoming trace stream. This bit is cleared when trace capture is enabled again after having stopped. This bit is operational only in the Circular-buffer mode. In all other modes, this bit is always 0.This bit does not indicate that a trigger has been embedded in the formatted output trace data from the TMC. Trigger indication on the output trace stream is determined by the programming of the Formatter and Flush Control Register, FFCR, 0x304. 1 = HIGH 0 = LOW
0	X	Full: This bit can be used to help determine how much of the trace buffer contains valid data.Circular-buffer mode : This flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until the TraceCaptEn bit is cleared and set. Software-read-FIFO mode and Hardware-read-FIFO mode : This flag indicates that the current space in the trace memory is less than or equal to the value programmed in the BUFWM register (ie. Fill level >= MEM_SIZE - BUFWM). In the ETR configuration, if the TMC is programmed for Scatter-Gather operation, this bit indicates whether the Trace memory is currently full regardless of the value programmed in the BUFWM register. TMC Disabled (TraceCaptEn=0 and TMCReady=1): The Full bit retains its last value. If TraceCaptEn is 0 and a 1 is written to it, the Full bit is cleared.The FULL output from the TMC reflects the value of this register bit, except when the Integration Mode bit in the ITCTRL register, 0xF00, is set. 1 = HIGH 0 = LOW

42.18.6 CORESIGHT_ETF_HUGO_CXTMC_REGS_RWP_0

The value written to this register must be a byte-address aligned to the width of the trace memory data bus and to a frame boundary.

Offset: 0x30018 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	RWP: This value represents the location in trace memory that will be accessed on a subsequent write to the trace memory.

42.18.7 CORESIGHT_ETF_HUGO_CXTMC_REGS_RRD_0

Reading this register enables data to be read from the trace memory.

Offset: 0x30010 I Read/Write: RO I Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RRD: Reads return data from Trace RAM

42.18.8 CORESIGHT_ETF_HUGO_CXTMC_REGS_CTL_0

This register controls trace stream capture.

Offset: 0x30020 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	TraceCaptEn: Setting this bit to 1 enables the TMC to capture trace data. 1 = HIGH 0 = LOW

42.18.9 CORESIGHT_ETR_HUGO_CXTMC_REGS_CTL_0

This register controls trace stream capture.

Offset: 0x50020 I Read/Write: R/W I Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	TraceCaptEn: Setting this bit to 1 enables the TMC to capture trace data. 1 = HIGH 0 = LOW

Notice

The information provided in this specification is believed to be accurate and reliable as of the date provided. However, NVIDIA Corporation ("NVIDIA") does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This publication supersedes and replaces all other specifications for the product that may have been previously supplied.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and other changes to this specification, at any time and/or to discontinue any product or service without notice. Customer should obtain the latest relevant specification before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer. NVIDIA hereby expressly objects to applying any customer general terms and conditions with regard to the purchase of the NVIDIA product referenced in this specification.

NVIDIA products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on these specifications will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this specification. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this specification, or (ii) customer product designs.

No license, either express or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this specification. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this specification is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

ARM

ARM, AMBA and ARM Powered are registered trademarks of ARM Limited. Cortex, MPCore and Mali are trademarks of ARM Limited. All other brands or product names are the property of their respective holders. "ARM" is used to represent ARM Holdings plc; its operating company ARM Limited; and the regional subsidiaries ARM Inc.; ARM KK; ARM Korea Limited.; ARM Taiwan Limited; ARM France SAS; ARM Consulting (Shanghai) Co. Ltd.; ARM Germany GmbH; ARM Embedded Technologies Pvt. Ltd.; ARM Norway, AS and ARM Sweden AB

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA, the NVIDIA logo, and Tegra are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright